CoopRobo Documentation

Versão 1.0.0

Gabriel Araujo

15 dez., 2020

Contents:

Collaborative Robotics project

- *Robots*

CAPÍTULO 1

Aerial Robots

1.1 VR-01

Este trabalho busca o estudo, documentação e desenvolvimento de técnicas de controle cooperativo de veículos aéreos não tripulados a fim de possibilitar que três aeronaves de asa fixa voem, autonomamente, em formação de esquadrilha.

O uso de múltiplos veículos aéreos não tripulados em missões como vigilância, monitoramento e buscas pode torná-las mais rápidas e com maiores chances de sucesso. Para tal, é necessário que os VANTs comuniquem-se entre si e tenham uma estratégia de controle que determine o que cada um deles deve fazer. Esse projeto inclui três VANTs de asa fixa disponíveis no Laboratório de Robótica Aérea da Universidade de Brasília e deverão ser estudadas e implementadas técnicas de controle cooperativo no hardware e software embarcado.

O sistema completo é constituído por quatro sub-sistemas: três aeronaves e uma estação base. A comunicação entre tais sub-sistemas é realizada através de rádios de comunicação. Cada sub-sistema possui seu próprio modem de comunicação e tem a capacidade de comunicar-se diretamente com qualquer outro sub-sistema.

Cada aeronave possui dois componentes principais que merecem destaque. Um deles é o piloto automático, dispositivo responsável pela aquisição, condicionamento e processamento de sinais provenientes dos sensores da

Fig. 1: Aeronave Ranger EX Volatex RC

aeronave e pelo controle dos atuadores do avião. O outro é o computador embarcado, responsável pelo processamento de dados, controle dos aviões (controle individual de cada avião ou controle cooperativo entre os três aviões) e pela comunicação com os outros sub-sistemas por meio dos rádios de comunicação.

1.1.1 Piloto Automatico

Nota: Este topico consiste em um resumo das informações disponíveis na pagina Getting Started - PX4.

Introdução

O PX4 é um piloto automático profissional de código aberto, desenvolvido tanto para atividades industriais quanto pela classe acadêmica, sendo apoiado pela comunidade mundial ativa. O PX4 pode ser executado em várias placas controladoras de voo. Merecendo destaque os controladores de voo de *hardware* aberto da série PixHawk, executando o PX4 no NuttX OS[1].

Dadas as opções de placas controladoras de voo disponíveis no mercado, escolheu-se o Pixhawk1 como piloto automático por seu melhor custo beneficio para o projeto.

O Pixhawk atua em diversos tipos de veículos, desde drones de corrida, monitoramento e carga a veículos terrestres e submersíveis para os mais diversos fins. Nesta documentação, focaremos na aplicação em aeronaves, onde o Pixhawk atua como um controlador de voo de uso geral, responsável pela aquisição, condicionamento e processamento de sinais provenientes dos sensores da aeronave e pelo controle dos atuadores da aeronave.

Ele oferece um ambiente de desenvolvimento compatível com sistemas Unix e Linux, facilitando o desenvolvimento de aplicações de software. O sistema Pixhawk possui capacidade de *multithreading*[2], ou seja, pode executar várias tarefas simultaneamente sem que uma interfira na outra através do compartilhamento de recursos do processo. Além disso, ele possui funções de piloto automático integrado com logs detalhados de missões e comportamento de voo[3].

[1] PX4 Autopilot User Guide. docs.px4.io

[2] Para mais informações a respeito de Multithreading (arquitetura computacional).

[3] Eduardo Moura Cirilo Rocha. 2017. Desenvolvimento de um sistema com veículos aéreos não-tripulados autônomos, Universidade de Brasília, Brasil.

Conceitos Básicos

Este tópico apresenta alguns conceitos básicos a respeito de veículos aéreos não tripulados e o uso da plataforma PX4.

Veículo aéreo não tripulado (VANT ou *drone*)

Um VANT é todo e qualquer tipo de aeronave que pode ser controlada nos 3 eixos de liberdade e não necessita de pilotos embarcados para ser guiado, podendo ser controlado remotamente ou autonomamente.

O "cérebro" de um drone é chamado de piloto automático, um instrumento responsavel por controlar a trajetória de voo da aeronave. Referindo-se a VANTs, o piloto automático consiste em um *software* de controle de voo sendo executado em um *hardware* especifico para a mesma função.

Estação de Controle em Solo (ECS)

Uma **Estação de Controle em Solo** (ECS), do inglês *Ground Control Station* (GCS), é uma plataforma de controle, normalmente uma aplicação de *software* sendo executada em um computador em solo, que se comunica com os VANTs por telemetria sem fio e provê aos operadores humanos o controle das aeronaves.

A estação em solo entrega ao controlador diversos dados em tempo real sobre o desempenho e posição dos VANTs e pode até servir como um «cockpit virtual», fornecendo muitos dos mesmos instrumentos que um piloto teria caso estivesse pilotando um avião. Contudo, um *software* de Controle em solo é normalmente utilizado para o planejamento, envio das missões de voo e definição de parâmetros de voo.

Existem mais de dez estações de controle em solo diferentes. Na área de controle de VANTs, os principais controladores são Mission Planner, APM Planner 2, MAVProxy, QGroundControl e UgCS. Para Tablet/Smartphone, há Tower (DroidPlanner 3), MAVPilot, AndroPilot e SidePilot.[4]

Plataforma Dronecode

O PX4 faz parte da Dronecode Plataform, uma plataforma completa para desenvolvimento de drones, sob uma licença de código aberto a comunidade. Incluindo, entre outras, os controladores PX4, a estação de controle terrestre QGroundControl, o Dronecode SDK e o Dronecode Camera Manager.[5]

Sensores

Os sistemas baseados em PX4 utilizam diversos sensores para determinar o estado do veículo (sendo estes essenciais para a estabilização e para possibilitar o controle autônomo). Os estados do veículo incluem: posição, direção, velocidade, velocidade do ar, orientação (atitude), taxas de rotação em diferentes direções, nível da bateria, etc.

O sistema requer minimamente um giroscópio, acelerômetro, magnetômetro (bússola), barômetro e um sensor de velocidade do ar para o caso de asas fixas (caso do projeto). É necessário ainda um GPS ou outro sistema de posicionamento para ativar todos os modos automáticos e alguns modos assistidos.

Os controladores da série Pixhawk já vem com um conjunto mínimo de sensores incorporados. Sensores adicionais/externos podem ser conectados ao controlador.

GPS e bússola

O PX4 suporta vários receptores e bússolas (magnetômetros) do Sistema de Navegação Global por Satélite (Global Navigation Satellite System - GNSS). Além de suportar os Receptores GPS Real-time Kinematic (RTK), otimizando os sistemas de GPS a uma precisão em nível de centímetros.

[4] Choosing a Ground Station - Conter documentation. ardupilot.org
[5] Dronecode Platform, Basic Concepts, PX4 Autopilot User Guide. docs.px4.io

Nota: Os controladores da série Pixhawk incluem uma bússola *interna*, porém recomendamos o uso de um módulo externo de bússola + GPS (*compass/GPS*), sendo este montado o mais longe possível dos cabos de alimentação dos motores para reduzir a interferência eletromagnética.

O PX4 suporta a conecção de até 4 magnetômetros internos ou externos, embora apenas um seja realmente utilizado para orientação. O sistema escolhe de forma automática a melhor bússola disponível com base em sua prioridade (bússolas externas têm maior prioridade). Se a bússola principal vier a falhar durante o voo, o sistema seleciona a aproxima maior prioridade. Caso a falha ocorra antes do voo, o carregamento plano de voo será negado.

Mais informações e a lista de GPS/bússola suportados pode ser encontradas em GPS/Compass.

Velocidade do ar

Dica: Os sensores de velocidade do ar são altamente recomendados para o funcionamento seguro de um VANT asa fixa ou VTOL (*Vertical Take-Off and Landing* - Decolagem e Aterragem Vertical).

O voo de um VANT asa fixa depende da velocidade do ar, já que é este que garante sua sustentação em voo e não a velocidade em relação ao solo. O piloto automático não possui outros meios para detectar estol (perda de sustentação da aeronave em voo), por este motivo os sensores de velocidade do ar são muito importantes.

Mais informações e a lista de sensores de velocidade do ar suportados pode ser encontradas em Airspeed sensors.

Distância (telêmetros)

Os sensores de distância fornecem medição de distância em tempo real. Podendo ser óptico, quando baseado em um mecanismo de focalização, ou ultrassônico (ecotelêmetro ou telêmetro acústico), quando utiliza reflexões sonoras. Eles são utilizados para melhorar a precisão do pouso, prevenir colisões, acompanhar o terreno, aviso de limites de altura, etc.

O PX4 suporta uma grande variedade de sensores de distância, usando tecnologias diferentes e oferecendo suporte a diferentes recursos. Mais informações e a lista de sensores de distância suportados pode ser encontrada em Distance sensors.

Fluxo optico

O PX4Flow é uma câmera inteligente de fluxo óptico com um sensor de sonar embutido que pode rastrear movimentos. O PX4 combina os dados do sensor com as informações de outras fontes de posição (GPS, por exemplo) para fixar uma posição de forma mais precisa. Este sensor pode ser utilizado em ambientes fechados, quando não há sinal de GPS disponível.

A maior parte de suas aplicações é direcionada a aeronaves de asas rotativas.

Especificações do Pixhawk

- **Processador**
  - 32-bit ARM Cortex M4 core with FPU
  - 168 Mhz/256 KB RAM/2 MB Flash
  - 32-bit failsafe co-processor
- **Sensores**
  - MPU6000 as main accel and gyro

  - ST Micro 16-bit gyroscope
  - ST Micro 14-bit accelerometer/compass (magnetometer)
  - MEAS barometer
- **Power**
  - Ideal diode controller with automatic failover
  - Servo rail high-power (7 V) and high-current ready
  - All peripheral outputs over-current protected, all inputs ESD protected
- **Interface**
  - 5x UART serial ports, 1 high-power capable, 2 with HW flow control
  - Spektrum DSM/DSM2/DSM-X Satellite input
  - Futaba S.BUS input (output not yet implemented)
  - PPM sum signal
  - RSSI (PWM or voltage) input
  - I2C, SPI, 2x CAN, USB
  - 3.3V and 6.6V ADC inputs
- **Dimensões**
  - Weight 38 g (1.3 oz)
  - Width 50 mm (2.0")
  - Height 15.5 mm (.6")
  - Length 81.5 mm (3.2")
- **Itens inclusos**
  - 1 x SanDisk Ultra micro SD Card (8GB)
  - 1 x MRC0225- Cable [3-Pins DF-13] to Switch+LED
  - 1 x MRC0224- Cable [2-Pins DF-13] to Buzzer
  - 1 x I2C Splitter
  - 2 x MRC0213- Cable [6-Pins JST-GH] to [6-Pins DF-13], (Telemetry Radio, Power module and Extra)
  - 1 x MRC0216- Cable [6-Pins DF-13] to [6-Pins DF-13], (For legacy products)
  - 4 x Damping Foams
  - 3 x Decals «APM Rover», «APM Copter» and «APM Plane»

1.1.2 Configuração do Piloto Automatico

A preparação do Pixhawk para voo consiste principalmente em instalar o firmware no dispositivo, conectar e calibrar os sensores e montar na estrutura da aeronave.

Esta seção contém os principais tópicos de configuração e montagem:

QGroundControl

O QGroundControl é uma das principais Estações de Controle em Solo (ECS) disponíveis atualmente a trabalhar com pilotos automáticos compatíveis com MAVLink, incluindo o PX4 e ArduPilot.

Para aplicar neste projeto, escolheu-se o QGroundControl como ECS por fornece uso fácil e direto para iniciantes, além de oferecer, a usuários experientes, suporte a recursos avançados no controle completo de voo e configurações do veículo com PX4.

Além disso, QGroundControl é uma das ECSs mais estáveis em relação as outras, possui uma interface simples e eficiente e está disponível em diversos sistemas operacionais, como Windows, Mac OS X, Linux, Android e o iOS.

Requisitos do Sistema

O QGroundControl pode ser executado normalmente na maioria dos computadores modernos. Um computador com um i5 e pelo menos 8 GB de RAM terá bom desempenho em todos os aplicativos do programa. Para uma melhor experiência, é aconselhável ter o sistema operacional em sua última versão estável.

Instalação

- **Windows**
  - Instalação do QGroundControl nas versões de 64 bits do Windows Vista ou posterior:
    1. Efetue o download do instalador do QGroundControl.exe
    2. Clique duas vezes no executável para inicializar o instalador.
- **Mac OS X**
  - Instalação do QGroundControl no Mac OS 10.10 ou posterior:
    1. Efetue o download do QGroundControl.dmg
    2. Clique duas vezes no arquivo .dmg para instalar
    3. Mova o aplicativo $ QGroundControl para a pasta *Application*
- **Ubuntu**

O Ubuntu possui um gerenciador de modem serial (*serial modem manager*) que interfere nas aplicações envolvendo robótica que utilizam uma porta serial (ou serial USB). Antes da instalação do *QGroundControl* é necessário remover tal gerenciador de modens e conceder ao seu usuário as permissões para acessar a porta serial. Também é preciso instalar o GStreamer para possibilitar o streaming de video.

- Antes de instalar o QGroundControl pela primeira vez:

1. Digite no prompt de comando:

```
sudo usermod -a -G dialout $USER
sudo apt-get remove modemmanager -y
sudo apt install gstreamer1.0-plugins-bad gstreamer1.0-libav -y
```

2. Execute logout e login novamente para ativar a alteração nas permissões de usuario.

- Instalação do QGroundControl no Ubuntu Linux 16.04 LTS ou posterior:

1. Efetue o download do QGroundControl.AppImage
2. Instale e execute os comandos do terminal:

```
cd Downloads
chmod +x ./QGroundControl.AppImage
./QGroundControl.AppImage
```

Dica: A última linha de comando não é necessário se o usuário for ao gerenciador de arquivos, procurar o arquivo QGroundControl baixado e clicar duas vezes nele.

- **Android**
  - O QGroundControl está disponível no Google Play Store em QGroundControl - play.google.com .
- **IOS**
  - O QGroundControl está disponível na App Store.

Firmware

A instalação do *firmware* no hardware do controlador de voo pode ser efetuada de duas formas, pelo uso de um programa de Estação de Controle Terrestre (ECT) ou diretamente pelo uso de ferramentas de desenvolvedor sem o uso de um programa auxiliar[1].

Dica: Antes de iniciar esta seção, recomenda-se o download e instalação do QGroundControl em seu computador.

Nota: A documentação oficial do QGroundControl está disponível em QGroundControl.

Instalação

Recomenda-se a instalação da versão mais recente do PX4, a fim de obter as correções de bug e as melhores e mais recentes funções.

Nota: Antes de instalar o firmware, todas as conexões USB do veículo devem ser desconectadas e o veículo não deve ser alimentado por uma bateria.

1. Selecione o icone da engrenagem (**Vehicle Setup**) na barra de ferramentas superior e, em seguida, selecione **Firmware** na barra lateral.
2. Conecte o Pixhawk diretamente ao seu computador via USB (não conecte através de um hub USB).
3. Selecione a opção **PX4 Flight Stack X.x.x Release** para instalar a versão mais recente do PX4 em seu controlador de voo (detectado automaticamente).
4. Clique em **OK** para iniciar a instalação.

O firmware seguirá com várias etapas de atualização (download do novo firmware, exclusão da versão antiga, etc.). O progresso geral é exibido em uma barra de progresso.

Assim que a instalação acabar e o firmware estiver instalado, o controlador será reiniciado e reconectado.

Mais Informações

- PX4 user guide > Firmware.
- QGroundControl user guide > Firmware.
- PX4 Setup Video (Youtube)

[1] Eduardo Moura Cirilo Rocha. 2017. Desenvolvimento de um sistema com veículos aéreos não-tripulados autônomos, Universidade de Brasília, Brasil.

Estrutura da aeronave

Após a instalação do firmware, é necessário configurar os parâmetros do firmware para a estrutura específica do seu veículo.

Definindo a estrutura

1. Com o QGroungControl aberto e o controlador conectado ao computador, selecione o ícone com 3 engrenagens (**Vehicle Setup**) na barra de ferramentas superior e, em seguida, selecione **AirFrame** na barra lateral.
2. Selecione o grupo de veículos que corresponde à estrutura da aeronave e, em seguida, utilize o menu suspenso dentro do grupo para escolher o tipo de estrutura que melhor corresponde ao seu veículo.
3. Clique em **Aplly and Restart**. Em seguida, vá em **Apply** no *prompt* para salvar as configurações e reiniciar o veículo.

Mais informações

- PX4 user guide > Airframe.
- QGroundControl user guide > Airframe.

Conexões

A imagem abaixo apresenta as conexões dos sensores e demais itens inclusos no Pixhawk. Cada parte será analisada com mais detalhes nas seções a seguir.

Campainha e interruptor de segurança

A campainha fornece sinais sonoros que indicam a situação do VANT. Enquanto o interruptor atua na segurança da aeronave, bloqueando e desbloqueando os motores.

Nota: O interruptor de segurança é ativado por padrão e quando ativado, não permite o voo, bloqueando os motores. Para desativar o modo de segurança, pressione e segure o interruptor por 1 segundo. Você pode ativar o modo de segurança novamente pressionando o interruptor.

Para conectar a campainha e o interruptor de segurança (itens obrigatórios), basta liga-lós ao Pixhawk como mostrado abaixo.

Divisor I2C

O *slitter* I2C expande a quantidade de portas I2C permitindo a conexão de até quatro periféricos ao Piwhawk. Utilize um cabo de 4 fios para conectar o *slitter* I2C e para alimentar uma bússola externa, um display LED, um sensor de velocidade do ar digital e/ou qualquer outro periférico compatível ao veículo.

Sensor de velocidade do ar

Em edição...

GPS + Compass

O GPS, outro dispositivo obrigatório, deve ser conectado à porta GPS (6 pinos) usando o cabo de 6 fios fornecidos no kit. A conexão da bússola é opcional, porém recomendamos fortemente sua utilização. Para conecta-lá, ligue um cabo de 4 fios a uma porta I2C do *slitter* I2C, como mostrado abaixo.

Nota: O GPS/bússola deve ser montado no chassi da aeronave o mais longe possível de outros componentes eletrônicos, com a seta indicadora voltada para a frente e o mais alinhada possível com o Pixhawk.

Rádio controle

O sistema de rádio controle (RC) é necessário caso deseje controlar manualmente seu veículo, dado que o Pixhawk não requer um sistema de rádio para modos de voo autônomo.

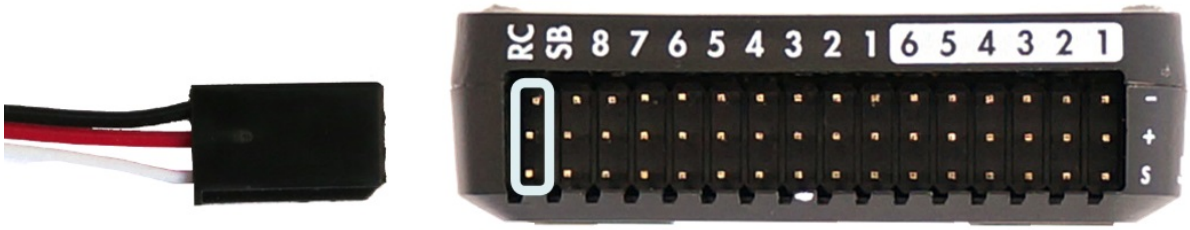
Para conectar o sistema de rádio controle, será necessário selecionar um transmissor/receptor compatível e depois vinculá-lo para que eles se comuniquem.

Dica: Leia as instruções que acompanham seu transmissor/receptor.

As instruções a seguir mostram como conectar os diferentes tipos de receptores ao Pixhawk:

- Os receptores Spektrum e DSM se conectam à entrada SPKT/DSM .

- Os receptores PPM-SUM e S.BUS conectam-se aos pinos de terra, potência e sinal RC, conforme mostrado.

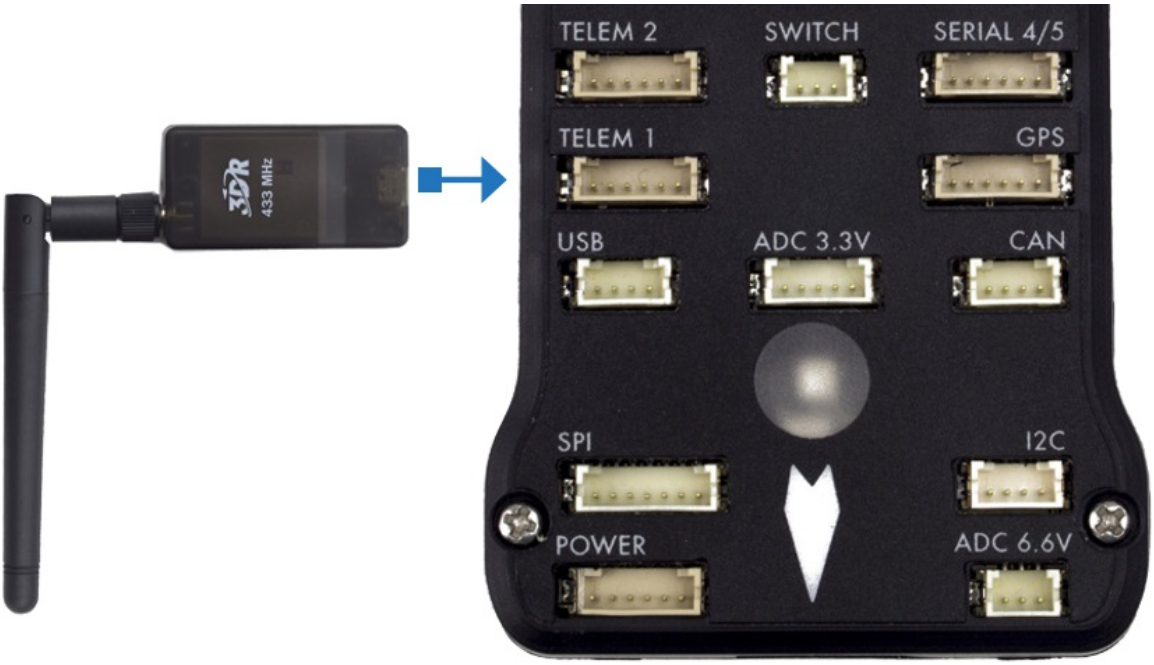


- Os receptores PPM e PWM que possuem um fio individual para cada canal devem se conectar à porta RC por meio de um codificador PPM (os receptores PPM-Sum usam um único fio de sinal para todos os canais).

Para obter mais informações sobre a seleção de um sistema de rádio, a compatibilidade do receptor e a ligação do seu par transmissor e receptor, consulte: Transmissores e receptores de controle remoto.

Telemetria

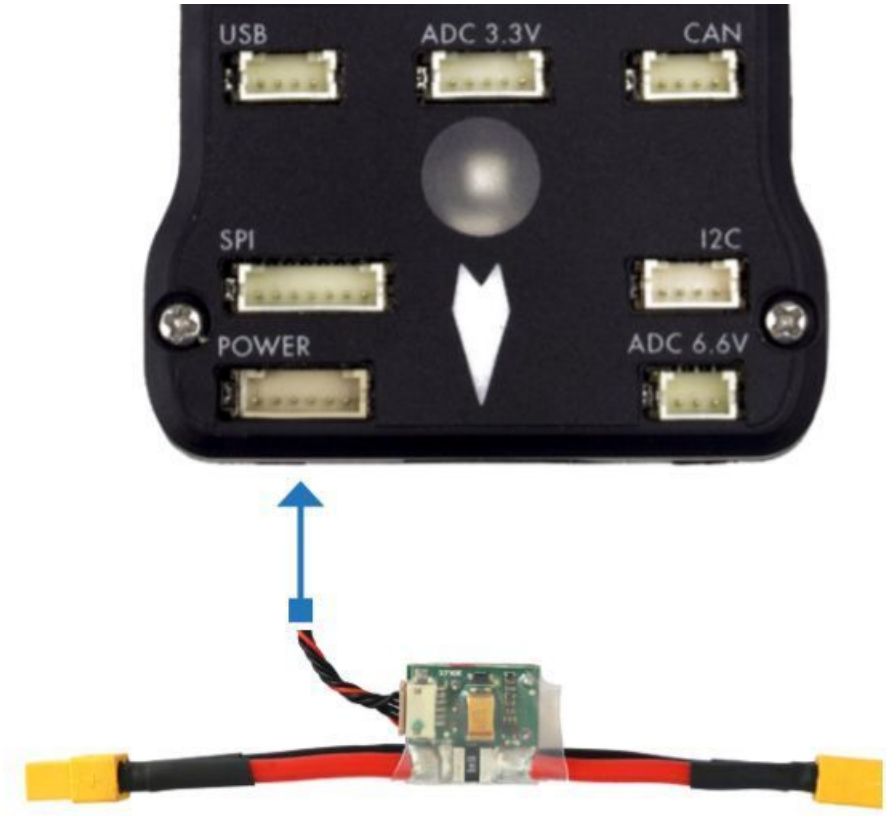
Os modens de telemetria podem ser usados para comunicar e controlar um veículo em voo a partir de uma estação terrestre (por exemplo, você pode direcionar o VANT para uma posição específica ou carregar uma nova missão). Um modem deve ser conectado ao seu veículo, como mostrado abaixo. O outro modem deverá ser conectado ao computador da estação terrestre ou dispositivo móvel (geralmente por uma porta USB).



Módulo de energia

O módulo de energia (*Power module* - PM) fornece energia ao controlador de voo da bateria e também envia informações sobre a corrente analógica e a tensão fornecida pelo módulo (incluindo a energia do controlador de voo e dos motores, etc.).

A saída do modulo de energia (PM) deve ser conectada à porta **POWER** do Pixhawk usando um cabo de 6 fios, como apresentado na imagem. A entrada do modulo deverá ser conectada a uma bateria de LiPo, enquanto a saída principal será responsável por fornecer energia aos ESCs e motores da aeronave (possivelmente através de uma placa de distribuição de energia, a depender da aeronave).



Sensor de distancia

O Pixhawk suporta vários sensores de distância diferentes, incluindo os Lidars (que usam lasers ou raios infravermelhos para medições de distância) e Sonars (que utilizam som ultrassônico), e também incluem os buscadores de alcance LED Maxbotix Sonar e Pulsed Light. Dessa forma, a instalação varia de dispositivo para dispositivo. Mais informações a respeito da configuração dos sensores pode ser visualizada em Rangefinders.

Fig. 2: Exemplo de alguns sensores de distância compatíveis

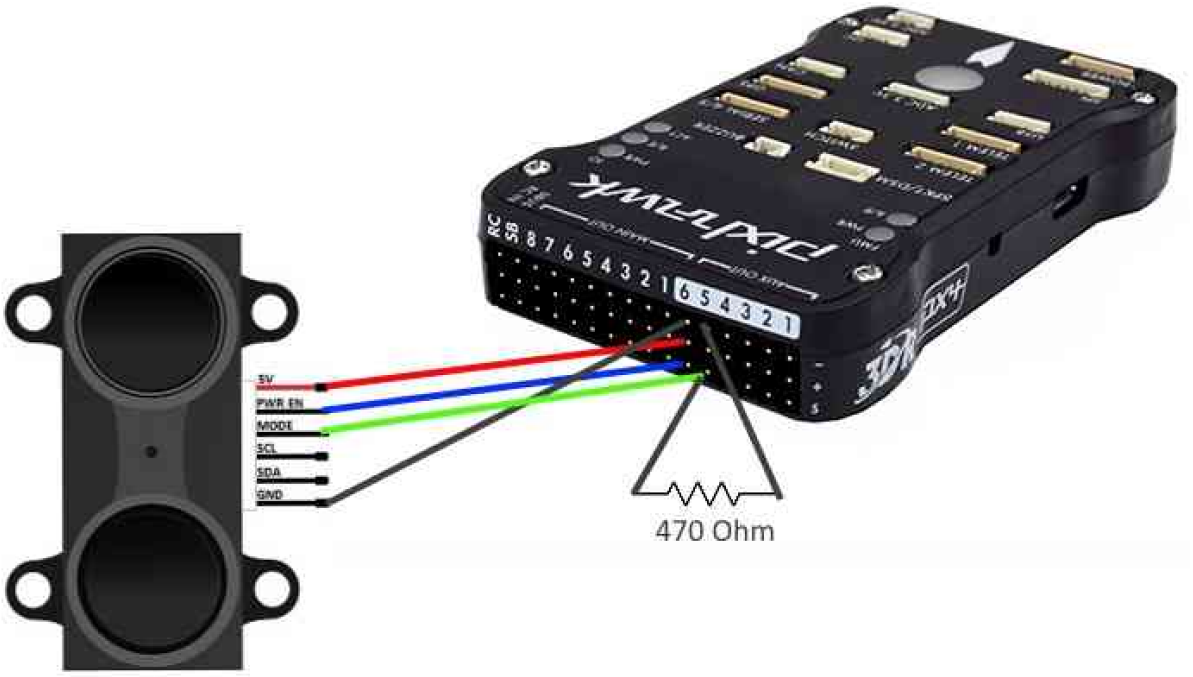
Para implementar o projeto, escolheu-se o sensor Lidar para habilitar a função de pouso automático devido sua maior precisão em relação aos demais. O sensor lidar pode ser conectado ao Pixhawk de duas formas, através do protocolo I2C na porta I2C (ou I2C *slitter*) ou por *pulse-width-modulation* (PWM) na trilha PWM.

De acordo com a documentação do Pixhawk, o lidar utilizado apresenta problemas de interferência com outros dispositivos quando conectado na porta I2C. Assim, escolheu-se a conexão por PWM. Um diagrama de conexão pode ser vista na tabela abaixo e o esquema de montagem pode ser visto na figura a seguir, onde o valor do resistor pode variar entre 200Ω e 1kΩ[1].

[1] Eduardo Moura Cirilo Rocha. 2017. Desenvolvimento de um sistema com veículos aéreos não-tripulados autônomos, Universidade de Brasília, Brasil

Table 1: Diagrama de conexão entre o Lidar e o Pixhawk

Sinal LIDAR-Lite	Sinal Pixhawk
J1	CH6 Out - V+
J2	CH6 Out - Signal (sinal interno 55)
J3	CH5 Out - Signal (sinal interno 54)
J4	
J5	
J6	Ch6 Out - Ground



Mais detalhes sobre a conexão podem ser encontrados em LIDAR-Lite Rangefinder.

Mais informações e referências

- Pixhawk Wiring Quick Start - PX4 User Guide
- Basic Assembly - PX4 User Guide
- Pixhawk Series - PX4 User Guide
- Peripheral Hardware - Ardupilot Docs

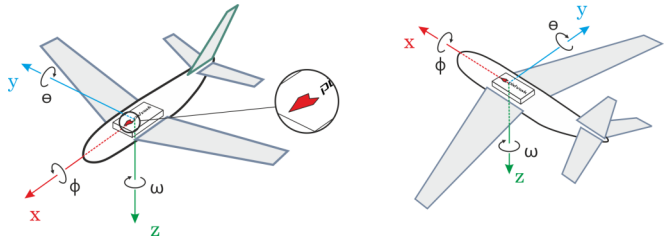
Montando o Pixhawk

Orientação do Piloto Automático

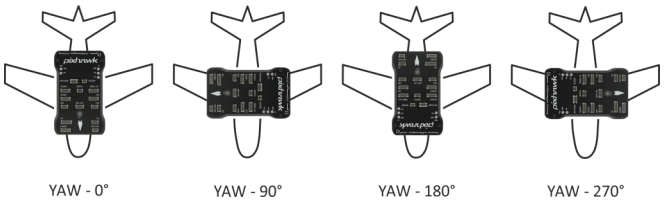
Por padrão, o controlador de voo e a bússola externa devem ser colocados na estrutura da aeronave orientados de modo que a seta aponte para a frente do veículo. Se a placa ou a bússola externa estiverem em qualquer outra orientação, será necessário corrigir a orientação no firmware.

Cálculo da Orientação

As compensações dos ângulos de rotação YAW, PITCH e / ou ROLL são calculados em relação à orientação vertical apontando para a frente (rotação no sentido horário em torno dos eixos Z, Y e X, respectivamente). Esse diagrama é chamado de *body frame* (diagrama de corpo) e a orientação padrão é dada por `ROTATION_NOME`.



Por exemplo, a imagem abaixo apresenta rotações de aeronaves em torno do eixo z (YAW), correspondendo, respectivamente, a: `ROTATION_NONE`, `ROTATION_YAW_90`, `ROTATION_YAW_180`, `ROTATION_YAW_270`.



Definindo a Orientação

Para definir as orientações no firmware:

Nota: Antes de definir a orientação, o QGroundControl deve ser iniciado, conectado ao veículo e o firmware já deve ter sido instalado na placa controladora de voo.

---

1. Selecione o ícone de **engrenagem** (Configuração do veículo) na barra de ferramentas superior e, em seguida, **Sensors** na barra lateral.
2. Selecione o botão **Set Orientations**.
3. Selecione a orientação do piloto automático, conforme calculado.
4. Selecione a **External Compass Orientation** (Orientação da bússola externa) da mesma maneira (esta opção será exibida apenas se o seu veículo tiver uma bússola externa).
5. Pressione **OK**.

Dica: A documentação completa sobre como ajustar a orientação do piloto automático está disponível em Orientação do piloto automático.

Isolamento de Vibrações

As placas Pixhawk possuem acelerômetros e giroscópios embutidos, sendo sensíveis a vibrações. Elevados níveis de vibração podem causar uma série de problemas, incluindo redução do desempenho de voo, voos mais curtos e maior desgaste do veículo. Em casos extremos, a vibração pode levar a falhas dos sensores, resultando em falhas de estimativa ou até mesmo a interrupção do voo.

Por essa razão, o Pixhawk vem com *espumas de amortecimento de vibrações*.

O Pixhawk deve ser montado na aeronave utilizando as espumas antivibratórias incluídas no kit. Ele deve ser posicionado o mais próximo possível do centro de gravidade do veículo.

Dica: Para determinar se os níveis de vibração estão muito altos e utilizar algumas técnicas para melhorar as características de vibração, recomenda-se o tópico PX4 user guide > Vibration Isolation.

Mais Informações

- Advanced Orientation Tuning.
- PX4 user guide > Sensor Orientation.
- QGroundControl user guide > Sensors.
- PX4 user guide > Vibration Isolation.

Calibração do Piloto Automático

Esta seção contém os principais tópicos de calibração do piloto automático:

Calibração da Bússola

Todos os magnetômetros internos e externos conectados ao Pixhawk serão configurados no processo de calibração da bússola

Etapas da Calibração

1. Abra o aplicativo QGroundControl e conecte o veículo pelo fio ao usb do computador.
2. Selecione o ícone **Engrenagem** (Configuração do veículo) na barra de ferramentas superior e, em seguida, **Sensores** na barra lateral.
3. Clique no botão do sensor da **Bússola**.
4. Clique **OK** para começar a calibração.
5. Coloque o veículo em umas das posições indicadas em vermelho (não calibrada), mantenha essa posição escolhida por alguns instantes. Uma vez solicitado (a imagem de orientação fica amarela), gire o veículo em torno do eixo especificado nas duas direções. Quando a calibração estiver concluída para a orientação atual, a imagem associada na tela ficará verde.
6. Repita o processo de calibração para todas as orientações do veículo.

Mais Informações

- PX4 user guide > Firmware.
- QGroundControl user guide > Sensors.
- PX4 Setup Video (Youtube)

Calibração do Giroscópio

Pelo aplicativo QGroundControl será orientado a posicionar o veículo em uma local com uma superfície plana e mantê-lo imóvel na posiçao determinada.

Etapas da Calibração

1. Clique no botão do sensor do **Giroscópio**.
2. Posicione o veículo sobre a superfície plana e mantenha-o na posição.
3. Clique **OK** para começar a calibração.
4. Quando finalizado, o aplicativo QGroundControl mostrará uma barra verde completa e uma contorno verde ao redor da imagem do veículo.

Dica: Caso ocorra algum movimento no veículo, o aplicativo QGroundControl irá automaticamente reiniciar o processo de calibração do giroscópio.

Mais informações

- QGroundControl user guide > Gyroscope.

Calibração do Acelerometro

O aplicativo QGroundControl mostrará todos os passos para colocar e segurar o veículo em diversas orientações.

Etapas da Calibração

Dica: É necessário ter a orientação do piloto automático definida para prosseguir com as etapas de calibração do acelerômetro. Caso contrário, entre na aba **Mountig the Pixhawk** e execute as etapas na subaba **Setting the Orientation**.

---

1. Abra o aplicativo QGroundControl e conecte o veículo pelo fio ao usb do computador.
2. Selecione o ícone **Engrenagem** (Configuração do veículo) na barra de ferramentas superior e, em seguida, **Sensores** na barra lateral.
3. Clique no botão do sensor do **Acelerometro**.
4. Clique **OK** para começar a calibração.
5. Coloque o veículo nas posições mostradas pela imagem que aparece na tela do aplicativo. Quando a posição é solicitada, a imagem de orientação fica amarela, deixe-a nessa posição (sem mover o veículo) até que a calibração seja concluída. A imagem ficará verde quando a calibração estiver concluída.
6. Repita o processo de calibração para todas as orientações do veículo.

Mais Informações

- PX4 user guide > Accelerometer.
- QGroundControl user guide > Sensors.
- PX4 Setup Video (Youtube)

Calibragem do Tubo de Pito

A calibração da velocidade do ar precisa ler uma linha de base estável com 0 velocidade do ar para determinar um deslocamento. Coloque as mãos sobre o pitot para bloquear qualquer vento (se não for necessário calibrar o sensor em ambientes fechados) e sopre o tubo com a boca (para sinalizar a conclusão da calibração).

Etapas da Calibração

1. Abra o aplicativo QGroundControl e conecte o veículo pelo fio ao usb do computador.
2. Selecione o ícone **Engrenagem** (Configuração do veículo) na barra de ferramentas superior e, em seguida, **Sensores** na barra lateral.
3. Clique no botão do sensor do **AirSpeed**.
4. Proteja o sensor do vento (ou seja, cubra-o com a mão). Tome cuidado para não bloquear nenhum dos orifícios.
5. Clique **OK** para começar a calibração.
6. Sopre na ponta do tubo pitot para indicar que a calibração está concluída.
7. Aguarde dois a três segundos antes de remover a proteção contra o vento, lembrando que a calibração será concluída em alguns segundos.

Dica: Soprar no tubo também é uma verificação básica de que as portas estáticas e dinâmicas estão instaladas corretamente. Se eles forem trocados, o sensor exibirá uma grande pressão diferencial negativa quando você soprar no tubo, e a calibração será interrompida com um erro.

Mais Informações

- QGroundControl user guide > AirSpeed.

Calibração do Radio

A opção Radio Setup, presente no QGroundControl, é usada para configurar o mapeamento dos principais bastões de controle de atitude da unidade de controle remoto (rotation, pitch, yaw, throttle) para os canais e calibrar as configurações mínimas, máximas, de compensação e reversão para todos os outros controles / canais RC.

Conexão Radio-Receptor-Transmissor

Antes de calibrar o sistema de rádio, o receptor e o transmissor devem estar conectados. O processo de conexão de um transmissor e receptor é específico do hardware.

- QGroundControl user guide > Spektrum receiver.
- FrSky receiver (Youtube)

Etapas de Calibração

No processo de calibração, será indicado que os bastões são movidos em um padrão específico que é mostrado no diagrama do transmissor no canto superior direito da tela.

1. Ligue o transmissor RC.
2. Abra o aplicativo QGroundControl e conecte o veículo pelo fio ao usb do computador.

3. Selecione o ícone **Engrenagem** (Configuração do veículo) na barra de ferramentas superior e, em seguida, **Rádio** na barra lateral.
4. Pressione **OK** para iniciar a calibração.
5. Defina o botão de opção do modo de transmissor que corresponde ao seu transmissor (isso garante que o QGroundControl exiba as posições corretas dos bastões para você seguir durante a calibração).
6. Mova os bastões para as posições indicadas no texto (e na imagem do transmissor). Pressione **Next** quando os bastões estiverem na posição. Repita para todas as posições.
7. Quando solicitado, mova todos os outros por toda as possibilidades de posições (você poderá observá-los se movendo no Monitor de canal).
8. Pressione **Avançar** para salvar as configurações.

Mais Informações

- Radio Setup Video (Youtube)

Pouso Automático

Após a conexão do Lidar ao sistema via PWM, alguns parâmetros do piloto automático devem ser alterados para que ele reconheça o sensor. Esses parâmetros podem ser facilmente alterados através do QGroundControl. São eles:

- RNG_FND = 5, indica que a conexão ocorre via PWM.
- RNDFND_MAX_CM = 4000, representa a distância máxima em que o sensor é confiável.
- RNDFND_STOP_PIN = 55, indica o pino conectado ao sinal de ativação do Lidar. Permite que o dispositivo reinicie o sensor caso ele para de fornecer dados.
- Os parâmetros RNDFND_SCALING e RNDFND_OFFSET devem ser ajustados de forma a se calibrar o sensor (costumam ser aproximadamente 0 e 1, respectivamente).

O sensor pode ser testado pelo QGroundControl, onde as leituras podem ser observadas no campo **Sonar Range**. Após a configuração do sensor, o piloto automático será capaz de pousar a aeronave de forma muito mais rápida e precisa. O pouso ocorre pelo envio do comando **Land** ao controlador, mas para que ele ocorra corretamente deve-se definir a posição da pista de pouso e deve-se ajustar os parâmetros de pouso, como, por exemplo, a velocidade com que o avião deve pousar.

A documentação detalhada sobre pousos automáticos pode ser encontrada em Automatic Landing.

1.1.3 Sistema Embarcado

Um computador embarcado deverá adicionado à cada aeronave de modo a comunicar-se com o piloto automático Pixhawk e com os computadores embarcados nas demais aeronaves através de um modem responsável por receber e transmitir informações. O microprocessador deverá processar as informações obtidas através do modem e repassar instruções ao piloto automático Pixhawk.

Computadores embarcados são sistemas microprocessados digitais no qual o computador é inteiramente encapsulado ou dedicado ao sistema que ele controla. Diferentemente de computadores de finalidade geral, como o computador pessoal, um sistema embarcado realiza um conjunto de tarefas predefinidas, geralmente com requisitos específicos. A predefinição das atividades a serem realizadas permite que os computadores embarcados possuam menor tamanho, peso, preço e capacidade de processamento do que computadores comuns que desempenham funções similares.

Entretanto, existem desvantagens em sua aplicação devido o mesmo motivo que os tornam atrativos. Por serem bem específicos geralmente estes dispositivos não podem ter sua função inicial alterada sem mudar boa parte de seu software ou estrutura do hardware, em alguns casos estes dispositivos chegam a não apresentar interface com o usuário.

Neste capítulo será explicitado os primeiros passos e procedimentos que tiveram de ser realizados para operar o microprocessador.

Computador Embarcado

Overo WaterStorm

Após considerar as aplicações mais específicas a serem realizadas, o computador embarcado escolhido foi o Overo WaterStorm Computer-On-Module (COM), esse sistema embarcado apresenta um processador DM3730 com arquitetura ARM Cortex-A8 e clock de base do processador de até 1 GHz. Além disso, esse computador está acoplado a uma placa de expansão Tobi que acrescenta ao computador embarcado conexões do tipo display DVI, Ethernet, USB Host, USB OTG, USB console, áudio Stereo e um segmento com 40 pin-headers que podem ser utilizados para a mais diversas funções, como modulação PWM, GPIO, alimentação, conversão analógico digital e comunicação serial.

Fig. 3: Sistema Gumstix com computador Overo WaterSTORM, placa de expansão Tobi e câmera Caspa VL

Acoplou-se também ao sistema uma câmera Caspa VL, capaz de capturar imagens coloridas com dimensão de 752 x 480 pixels em uma frequência de 60 imagens por segundo. Esses três componentes são produzidos pela empresa Gumstix, fabricante de hardware especializada em computadores pequenos do tipo computador-em-módulo (COM - Computer-On-Module), muito utilizados para sistemas embarcados.

Apesar do tamanho pequeno, a combinação da Overo COM com a placa de extensão TOBI possui o mesmo desempenho do que um computador Linux completo de tamanho normal, maior do que outros sistemas desse tipo encontrados no mercado, como, por exemplo, o computador Raspberry Pi.

Especificações

- **Camera** - Camera Connector: 27-Pin (OMAP ISP)
- **Mechanical** - Length: 58 mm - Width: 17 mm

- **Memory** - Flash Memory (NAND): 512
- **Processor** - Graphics Acceleration: PowerVR SGX530 with OpenGL - Digital Signal Processor: C64x+ - Processor: Texas Instruments OMAP3730 - Processor Architecture: ARM Cortex-A8 - Processor Base Clock: 800 MHz - Processor Max Clock: 1 GHz
- **Power** - Power Management: Texas Instruments TPS65950
- **Storage** - Storage Expansion via microSD Card Slot

Nota: As especificações completas do computador, da placa de extensão e da câmera estão disponíveis nas fichas técnicas abaixo.

- `Datasheet – Overo Waterstorm COM`
- `Manual – Overo Waterstorm COM`
- `Datasheet – Placa de extensão TOBI`
- `Datasheet – Câmera Caspa VL`

Referências

- PITA, H. C. Desenvolvimento de sistema de comunicação multiplataforma para veículos aéreos de asa fixa. Faculdade de Tecnologia, Universidade de Brasília, 2018.
- Overo® WaterSTORM COM - Gumstix Store

Sistema Operacional

Um computador digital com determinada complexidade que exige o gerenciamento dos recursos do sistema e tais funções primárias necessitam de um sistema operacional. O núcleo ou kernel é a parte mais importante e de nível mais baixo de um sistema operacional, ele tem a função de definir qual programa recebe atenção do processador, gerenciar memória, criar um sistema de arquivos, gerenciar o sistema de comunicação etc.

O primeiro passo para a utilização desse computador, é a criação e configuração de uma imagem de sistema operacional que atende aos requisitos do projeto. São eles: compatibilidade com o computador utilizado, *Overo WaterStorm COM*, e suporte para aplicações em tempo real.

Um Sistema Operacional de Tempo Real ou RTOS (*Real Time Operating Systems*) é um sistema operacional destinado à execução de múltiplas tarefas com tempo de resposta a um evento (externo ou interno) pré-definido. Existem duas abordagens para a execução de aplicações de tempo real em Linux, uso de ferramentas que implementam um kernel duplo ou o uso de RTL (Real-time Linux).

RT-Mag

Inicialmente, foi decidido a utilização da ferramenta RT-MaG como sistema operacional do sistema embarcado.

O projeto RT-MaG (*Real-Time - Marseille Grenoble Project*) é um projeto desenvolvido pelo Gipsa-Lab (Grenoble, França) e o Institute of Mouvement Sciences (ISM, Marseille, França). O objetivo deste projeto é fornecer ferramentas eficientes para a prototipagem rápida de robôs para pesquisa e aplicações acadêmicas. O RT-MaG fornece uma caixa de ferramentas para Matlab e Simulink para programar sistemas Linux-COM. Com a ferramenta, pode-se facilmente gerar um aplicativo autônomo em tempo real a partir de um modelo Simulink para um robô usando um sistema Linux.

Essas ferramentas consistem em um conjunto de blocos simulink que fornecem acesso direto às entradas e saídas do computador. Os modelos Simulink são convertidos automaticamente em aplicações em tempo

real. O uso dessas ferramentas é totalmente gratuito. Além disso, atualmente, o Gumstix Overo COM é totalmente compatível com o sistema RT-MaG.

Entretanto, a ferramenta RT-MaG toma para si muitas das operações necessárias para a operação do nosso sistema, o que impossibilita utiliza-lo da maneira que ele foi idealizado, em consequência disto a demasiada simplificação da etapa poderia prejudicar aplicações futuras. Com essa ferramenta seria inviável utilizar o protocolo de comunicação *MAVLink* do piloto automático para comunicação entre os dispositivos ou aeronaves, por exemplo.

Destaca-se ainda a documentação desatualizada, que dificultou a instalação dos componentes da ferramenta como a toolbox do Matlab, que nunca chegou a funcionar, e o sistema operacional do computador embarcado. A complexidade na utilização do sistema aumentava a cada etapa enquanto mesmo as etapas iniciais mais simples ainda não funcionavam adequadamente.

Nota: Mais detalhes do projeto RT-MaG podem ser encontrados em Projet RT-MaG.

Linux

O Linux é um sistema operacional popularmente utilizado em sistemas embarcados. Além de fornecer suporte para mais arquiteturas computacionais que qualquer outro sistema, ele ainda é leve e possui código aberto, minimizando os custos de implementação. Dos diferentes sistemas operacionais suportados pelas placas Gumstix Overo, destacam-se os sistemas baseados em Linux. Sendo o **Ubuntu** e o **Yocto Project** os principais, além de serem recomendados pelo próprio fabricante.

Fig. 5: Tux, a mascote do Linux

Projeto Yocto

O projeto Yocto é um projeto de colaboração open source da Linux Foundation, cujo objetivo é produzir e fornecer metadados, ferramentas e processos para ajudar seus usuários a criar distribuições baseadas em Linux para *softwares* embarcados, independentemente da arquitetura do sistema.

Um elemento a ser destacado dentre os componentes do Projeto Yocto é o sistema de compilação baseado na arquitetura OpenEmbedded, que permite que os desenvolvedores criem suas próprias distribuições Linux especificas para seu ambiente, de acordo com suas próprias necessidades. Essas configurações do Project Yocto fornecidas pelos fornecedores de hardware geralmente incluem configurações do kernel, módulos do kernel, firmware do kernel e pacotes do sistema básico.

Outra ferramenta importante do Yocto Project é o sistema de compilação por referência Poky. Ele contém a ferramenta BitBake, que permite a compilação cruzada independentemente da plataforma. Além disso, o BitBake gerencia todos os arquivos de configuração e dados, e tenta reduzir o tempo de compilação usando todos os recursos de processamento disponíveis.

Infelizmente, com a ampla versatilidade do Projeto Yocto, a complexidade do processo de criação de uma distribuição personalizada também está aumentando.

Nota: Mais detalhes do projeto Yocto podem ser encontrados em yoctoproject.org.

Ubuntu

Ubuntu é um sistema operacional de código aberto, desenvolvido a partir do núcleo Linux, baseado no Debian. O Ubuntu é desenvolvido pela Canonical e pela comunidade em um modelo de governança meritocrática. A Canonical fornece atualizações gratuitas de segurança e suporte para cada versão do Ubuntu. Todas as versões são disponibilizadas sem custo algum.

A vantagem de se utilizar o sistema Ubuntu é que esse é um sistema operacional a partir do núcleo Linux muito difundido que já contém diversos softwares que podem ser úteis para algumas aplicações futuras, ele contém, por exemplo, um compilador o que facilita a criação e execução de códigos simples para testes rápidos.

A desvantagem de se utilizar este sistema operacional é que podem ser executadas muitas tarefas paralelas desnecessárias que diminuem a especificidade e o desempenho do computador embarcado.

Nota: Mais detalhes a respeito do Ubuntu podem ser encontrados em ubuntu.com.

Sistema Escolhido

Chegamos a instalar o RT-Mag no sistema embarcado, entretanto, devido a complicações posteriores à instalação do sistema operacional, optou-se por não mais utilizar essa ferramenta.

Decidiu-se então utilizar o núcleo oferecido pelo Projeto Yocto por ser específico para o modelo de computador embarcado. Optando pela instalação do sistema Ubuntu 15.04 em um dos computadores com o intuito de analisar as diferenças entre os dois sistemas operacionais e realizar testes.

Entretanto, o sistema Ubuntu, apesar de ser uma versão estável e adaptada para o sistema em questão, apresentou erros não solucionados no processo de instalação, impossibilitando a instalação do sistema em um cartão SD.

Referencias

- PITA, H. C. Desenvolvimento de sistema de comunicação multiplataforma para veículos aéreos de asa fixa. Faculdade de Tecnologia, Universidade de Brasília, 2018.
- ROCHA, E. M. C. Desenvolvimento de um sistema com veículos aéreos não-tripulados autônomos. Faculdade de Tecnologia, Universidade de Brasília, 2017.
- Phanuel Hieber. Yocto Project on the Gumstix Overo Board. Technische Universität München.
- RT-MaG Project - gipsa-lab.fr
- Yocto Project - yoctoproject.org

Instalando o Sistema Operacional

A instalação do sistema operacional não é uma tarefa trivial, além disso existe uma escassez de documentação detalhada e completa que explique como instalar o sistema operacional no computador embarcado, logo será documentado nesta seção os procedimentos necessários para a instalação de um sistema operacional. Na fase atual dos trabalhos instalamos ambos os sistemas, de tal forma, podemos decidir posteriormente qual dos dois sistemas será melhor para nossa aplicação.

Nota: Os tutoriais oficiais podem ser encontrados no site da Gumstix e nos repositórios do GitHub do projeto Yocto e Ubuntu para produtos Gumstix.

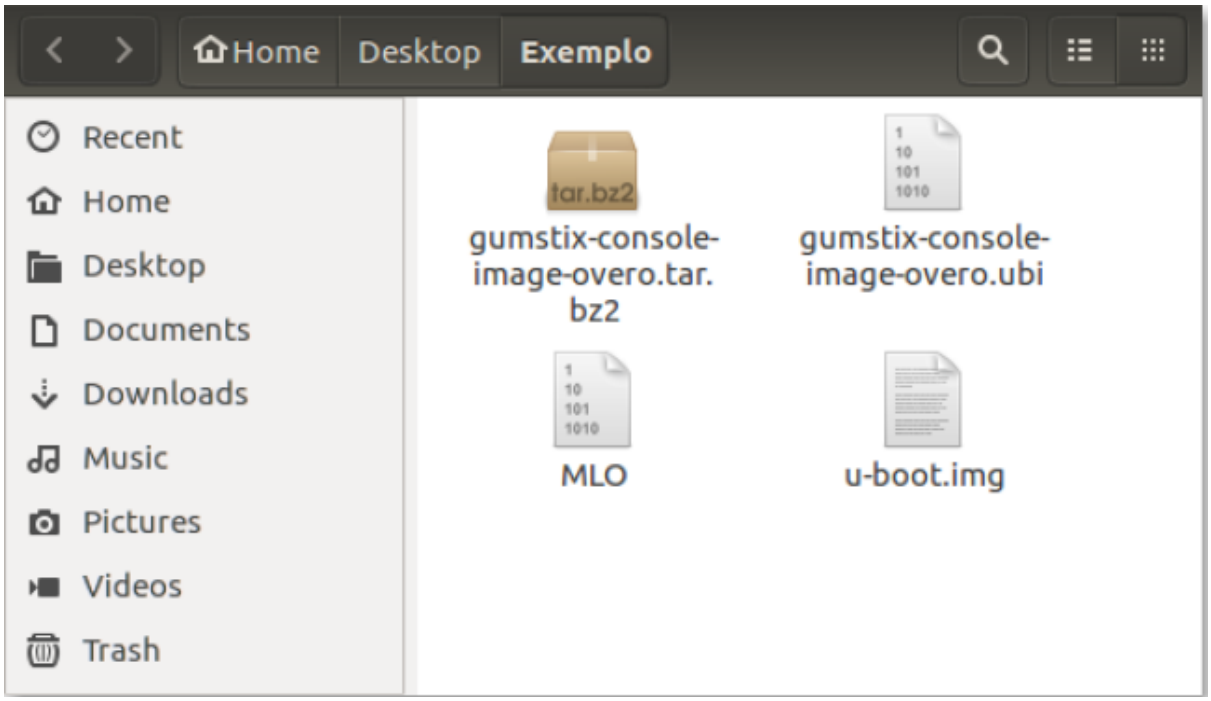
Obtenção das imagens do SO

Essencialmente o dispositivo precisa apenas executar um pequeno programa, geralmente localizado em uma memória não volátil do tipo *Read-Only Memory* (ROM), para acessar a outro dispositivo de memória não volátil que armazene o sistema operacional, e carregar o sistema operacional na memória volátil de rápido acesso ou *Random Access Memory* (RAM) onde ele poderá ser executado. Em sistemas mais robustos ocorre, na verdade, um encadeamento desses pequenos programas, chamados de *bootloaders*, onde um primeiro estágio executa um segundo estágio que carrega programas mais complexos e, por sua vez, executa um terceiro estágio e assim por diante até que o sistema operacional seja completamente carregado e esteja pronto para ser executado por si só.

Existem dois métodos para obter os sistemas operacionais para o Gumstix Overo. O primeiro método é baixar uma imagem pré-compilada diretamente do Gumstix. O segundo método é construir você mesmo a imagem em seu computador. A criação manual da imagem do sistema operacional possui benefícios adicionais, como personalização ou adição de pacotes binários adicionais à sua imagem base. A possibilidade de personalização será muito importante no desenvolvimento do projeto.

Obtenção das imagens do Yocto Project

Para realizar o processo de iniciação do sistema operacional Yocto Project no computador embarcado Gumstix, precisamos de três arquivos específicos, são eles o primeiro estágio do sistema de iniciação, o arquivo MLO (*Minimal Loader*), o segundo estágio do sistema de iniciação, o arquivo *u-boot.img* (a sigla vem de *Universal Bootloader*), e a imagem do sistema, que em nosso caso será o Yocto 1.8.2 com kernel Linux.



A figura mostra um exemplo dos arquivos descritos no parágrafo anterior, observe que, neste caso, há também uma pasta compactada que contém os arquivos raiz do sistema operacional. O modo mais simples encontrado para se obter esses arquivos e a imagem do sistema operacional é seguindo os passos do arquivo README.md do repositório do projeto Yocto para produtos Gumstix. A vantagem

de se utilizar esse método ao invés de simplesmente obter a imagem pronta do sistema operacional é que caso seja necessário poderemos modifica-la.

Esse tutorial explica como construir manualmente a imagem do sistema Yocto e realizar todos os procedimentos através de linhas de comando do terminal do Linux, com ênfase no **Ubuntu 14.04 (LTS)**. Porém, para executar essa etapa é altamente recomendado o cumprimento dos requisitos indicadas pelo projeto Yocto.

Requisitos do Sistema

Nota: Para mais informações a respeito dos requisitos do sistema, consulte System Requirements - Yocto Project Reference Manual.

O desenvolvimento de projetos no ambiente do Yocto Project requer que alguns requisitos sejam cumpridos, são eles:

- Um sistema com no mínimo 25 GB de espaço livre em disco executando uma distribuição Linux suportada. Se o sistema host suportar vários núcleos e encadeamentos, você poderá configurar o sistema de construção do Yocto Project para diminuir significativamente o tempo necessário para construir imagens.
- Pacotes apropriados instalados no sistema utilizado para realizar as compilações.
- Uma distribuição do Projeto Yocto.

Atualmente, o Project Yocto é suportado nas seguintes distribuições Linux.

- Ubuntu 12.04 (LTS)
- Ubuntu 13.10
- Ubuntu 14.04 (LTS)
- Fedora release 19 (Schrödinger's Cat)
- Fedora release 20 (Heisenbug)
- CentOS release 6.4
- CentOS release 6.5
- Debian GNU/Linux 7.0 (Wheezy)
- Debian GNU/Linux 7.1 (Wheezy)
- Debian GNU/Linux 7.2 (Wheezy)
- Debian GNU/Linux 7.3 (Wheezy)
- Debian GNU/Linux 7.4 (Wheezy)
- Debian GNU/Linux 7.5 (Wheezy)
- Debian GNU/Linux 7.6 (Wheezy)
- openSUSE 12.2
- openSUSE 12.3
- openSUSE 13.1

Nota: Para obter uma lista mais detalhada de distribuições que suportam o Projeto Yocto, consulte a seção Supported Linux Distributions em Yocto Project Reference Manual.

Para a construção da imagem do sistema operacional, o sistema de compilação deve possuir as seguintes versões do *softwares* Git, tar e Python.

- Git 1.8.3.1 ou posterior
- tar 1.27 ou posterior
- Python 3.4.0 ou posterior

Nota: Consulte a seção Required Git, tar, and Python Versions no Yocto Project Reference Manual para obter informações.

Além disso, recomenda-se atualizar os repositórios do Linux. Para tal, no caso de distribuição Ubuntu, basta executar o seguinte comando:

```
$ sudo apt-get update && sudo apt-get upgrade
```

É necessária ainda a instalação dos pacotes de host essenciais para a construção da imagem. O comando a seguir instala os pacotes de host com base em sistemas com distribuição Ubuntu.

```
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo␣
↪gcc-multilib build-essential chrpath socat libsdl1.2-dev xterm␣
↪curl
```

Nota: Para instalar os pacotes de host em outras distribuições Linux suportadas, consulte a seção Required Packages for the Build Host em Yocto Project Reference Manual.

Configurando a imagem

Nota: O sistema operacional utilizado para testar os comandos foi Ubuntu 14.04 (LTS).

Linhas de comando Linux para obtenção e montagem da imagem.

1. **Instalando o repositório**

Para fazer o download das imagens do Yocto, primeiro precisamos instalar o comando **repo**. Em resumo, o repo é basicamente um invólucro do git, que fornece uma maneira simples de agrupar vários repositórios git diferentes em um único projeto. Caso tenha interesse em mais informações sobre o comando **repo**, acesse repo - gerrit.googlesource.com.

Baixe os scripts do repositório

```
$ curl http://commondatastorage.googleapis.com/git-repo-downloads/
↪repo > repo
```

Torne os arquivos executáveis

```
$ chmod a+x repo
```

Mova os arquivos para o caminho do sistema

```
$ sudo mv repo /usr/local/bin/
```

Se tudo ocorrer bem, deverá aparecer uma mensagem de utilização similar a imagem ao executar o comando a seguir. Esse comando não é obrigatório.

```
$ repo --help
```

2. **Criando um repositório local**

Crie um diretório para os arquivos e altera o diretório de execução para o novo repositório

```
lucas@lucas-Ubuntu2-7348:~$ repo --help
repo: warning: Python 2 is no longer supported; Please upgrade to Python 3.6+.
usage: repo COMMAND [ARGS]

repo is not yet installed.  Use "repo init" to install it here.

The most commonly used repo commands are:

  init      Install repo in the current working directory
  help      Display detailed help on a command

For access to the full online help, install repo ("repo init").
```

```
$ mkdir yocto
$ cd yocto
```

Agora com o repositório já instalado, faremos o download de todas as configurações do Yocto para o nosso projeto. O comando **init** pode levar algum tempo, pois faz o download de todos os repositórios git associados ao projeto. Já o comando **-b** especifica a ramificação a ser usada e o comando **fido** seleciona o ramo mais estável do repositório.

```
$ repo init -u git://github.com/gumstix/yocto-manifest.git -b fido
```

Uma inicialização bem-sucedida terminará com uma mensagem informando que o **.repo** foi inicializado no seu diretório de trabalho. Agora seu diretório deve conter uma pasta *.repo* onde os arquivos de controle de repositório estão armazenados, mas não é necessário abrir o diretório.

```
lucas@lucas-Ubuntu2-7348:~/yocto$ repo init -u git://github.com/gumstix/yocto-manifest.git -b fido
repo: warning: Python 2 is no longer supported; Please upgrade to Python 3.6+.
Downloading Repo source from https://gerrit.googlesource.com/git-repo
remote: Finding sources: 100% (9/9)
remote: Total 9 (delta 0), reused 9 (delta 0)
Unpacking objects: 100% (9/9), done.
repo: Updating release signing keys to keyset ver 2.3
repo: warning: Python 2 is no longer supported; Please upgrade to Python 3.6+.
Downloading manifest from git://github.com/gumstix/yocto-manifest.git
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 534 (delta 6), reused 19 (delta 5), pack-reused 514

Your identity is: LucasCampos98 <[redacted]>
If you want to change this, please re-run 'repo init' with --config-name

Testing colorized output (for 'repo diff', 'repo status'):
  black     red       green     yellow    blue      magenta   cyan      white
  bold      dim       ul        reverse
Enable color display in this user account (y/N)? y

repo has been initialized in /home/lucas/yocto
```

3. **Baixando os arquivos**

O comando a seguir é usado para garantir que todos os seus repositórios estejam atualizados e é útil para atualizar suas configurações do Yocto se você fizer uma compilação posteriormente.

```
$ repo sync
```

Nota: Está etapa pode demorar mais de 20 minutos, dependendo da sua conexão de internet.

Force todos os arquivos temporários a serem escritos em dispositivos permanentes através do comando:

```
$ sync
```

4. **Iniciando o Yocto Project Build Environment**

Aviso: Se, por algum motivo, você cancelar a atividade antes de concluir a compilação do Yocto, será necessário executar este comando todas as vezes antes de seguir para as próximas etapas. Lembre-se de que isso também se aplica a compilações futuras.

Agora que temos nossas configurações básicas do Yocto, entraremos em nosso ambiente de compilação. Por meio do comando a seguir, iremos copiar as informações de configuração padrão no diretório **poky/build/conf** e configurar algumas variáveis de ambiente para o sistema de montagem da imagem.

```
$ export TEMPLATECONF=meta-gumstix-extras/conf
$ source ./poky/oe-init-build-env
```

Nota: Este diretório de configuração não está sob controle de revisão, portanto você pode editar esses arquivos de configuração para sua instalação específica.

---

5. **Criando a imagem**

O project Yocto utiliza o bitbake para compilar a imagem do Yocto Linux. O Bitbake basicamente compila apenas o SO, o kernel, os módulos e todos os pacotes incluídos no SO Linux de destino.

Dica: (**OPCIONAL**) Se você tiver familiaridade com a compilação via make, poderá acelerar o processo de compilação dizendo ao bitbake para compilar com mais threads. Esta etapa não é necessária, mas se você estiver compilando em um sistema com uma CPU de ponta com muitos núcleos, isso acelerará o tempo de compilação. Por exemplo:

```
$ export PARALLEL_MAKE="-j 8"
```

O número «8» indica a quantidade de núcleos a ser utilizada na compilação. **Vale ressaltar que você não deve especificar um valor -j maior que a quantidade de núcleos de CPU presentes em sua máquina de construção**.

Assim, para baixar os códigos fonte e compilar as imagens do sistema execute:

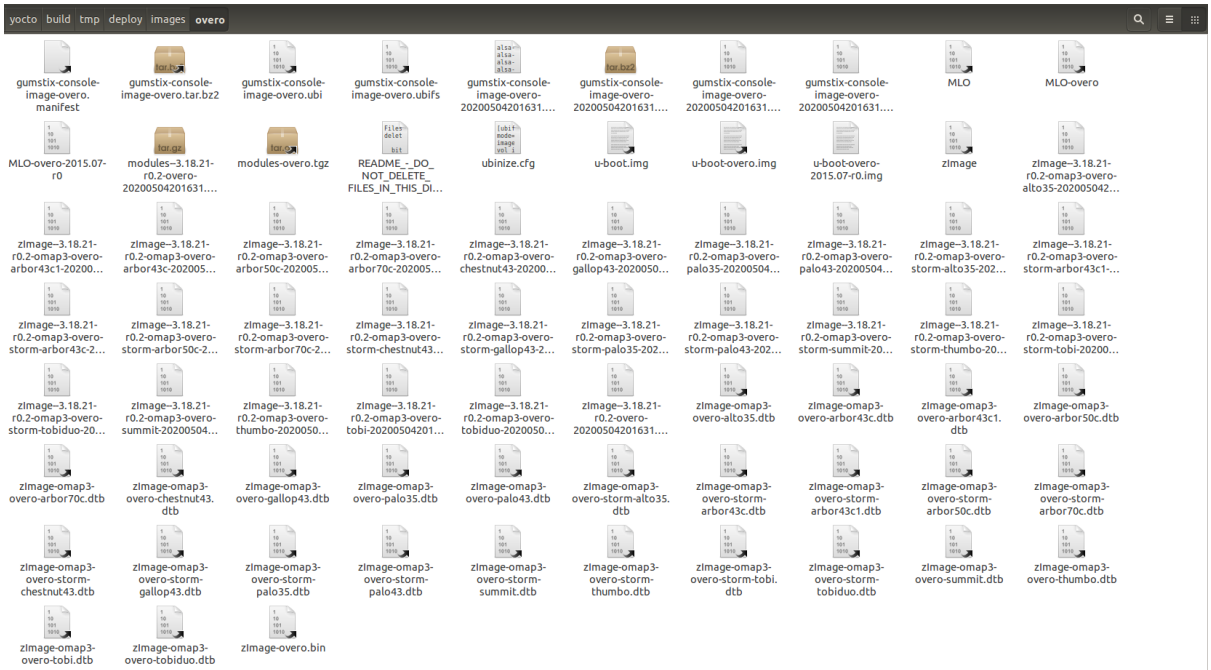
```
$ bitbake gumstix-console-image
```

Nota: Esse processo baixa vários gigabytes de código e, em seguida, faz uma enorme compilação. Portanto, certifique-se de ter pelo menos os 25GB de espaço livre. Esta etapa pode levar um dia ou mais para a criação da imagem, a depender da sua conexão de internet. Não se preocupe, é apenas a primeira compilação que demora um pouco.

Após a finalização da execução de todos os comandos, recomenda-se verificar a pasta **/yocto/build/tmp/deploy/images/overo**, essa pasta deve conter arquivos binários de kernel e bootloaders e arquivos de diretório raiz no formato .tar.

A figura abaixo apresenta um exemplo do conteúdo da pasta descrita, essa pasta deve ser semelhante ao obtido após a execução dos procedimentos anteriores.

Na figura podemos encontrar tanto os bootloaders necessários descritos anteriormente como o binário (.ubi) e arquivos do diretório raiz de algumas versões do projeto Yocto.



Aviso: Possíveis causas de falhas provavelmente estão relacionadas com softwares faltosos ou desatualizados, sistema operacional não compatível ou falta de espaço livre.

Referências

- PITA, H. C. Desenvolvimento de sistema de comunicação multiplataforma para veículos aéreos de asa fixa. Faculdade de Tecnologia, Universidade de Brasília, 2018.
- Gumstix Repo Manifests for the Yocto Project Build System - github.com
- Yocto Project Quick Start - yoctoproject.org
- Yocto Project Reference Manual - yoctoproject.org
- Building Yocto Linux Images for the Gumstix Overo - hackgnar.com

Preparando o Cartão de Memória

Uma vez obtida a imagem do sistema operacional podemos transferir os arquivos para o computador embarcado para, enfim, ligá-lo. Essa tarefa será realizada por meio de um cartão SD que funcionará como o disco rígido do computador embarcado. Logo, o cartão SD irá conter tanto os programas necessários para boot, que serão utilizados apenas na inicialização do computador, quanto os outros programas podem ser utilizados a qualquer momento e realizarão modificações constantes no cartão SD. Portanto a melhor maneira de lidar com essa divisão é particionar o cartão SD em duas partições que serão denominadas boot e rootfs.

O sistema de gestão de arquivos define o método que o sistema operacional irá utilizar para armazenar nos espaços de memória os arquivos e suas informações, ou metadados dos arquivos, como nome, espaço de memória ocupado, datas de alterações e últimos acessos. Existe uma grande variedade de sistemas de gestão de arquivos com as mais diversas complexidades. Mas o que podemos precisar nesse trabalho e em trabalhos futuros é o sistema «FAT», um sistema antigo geralmente utilizado em mídias e, normalmente, universal. Já o «ext» é um sistema elaborado especificamente para o Linux e não é possível acessá-lo por um outro sistema operacional sem um programa para essa finalidade.

Esse é um procedimento muito comum e existem inúmeras maneiras de fazê-lo, entretanto, aqui utilizaremos o próprio gerenciador de discos do Linux para realizar o particionamento, por ser uma ferramenta simples, intuitiva

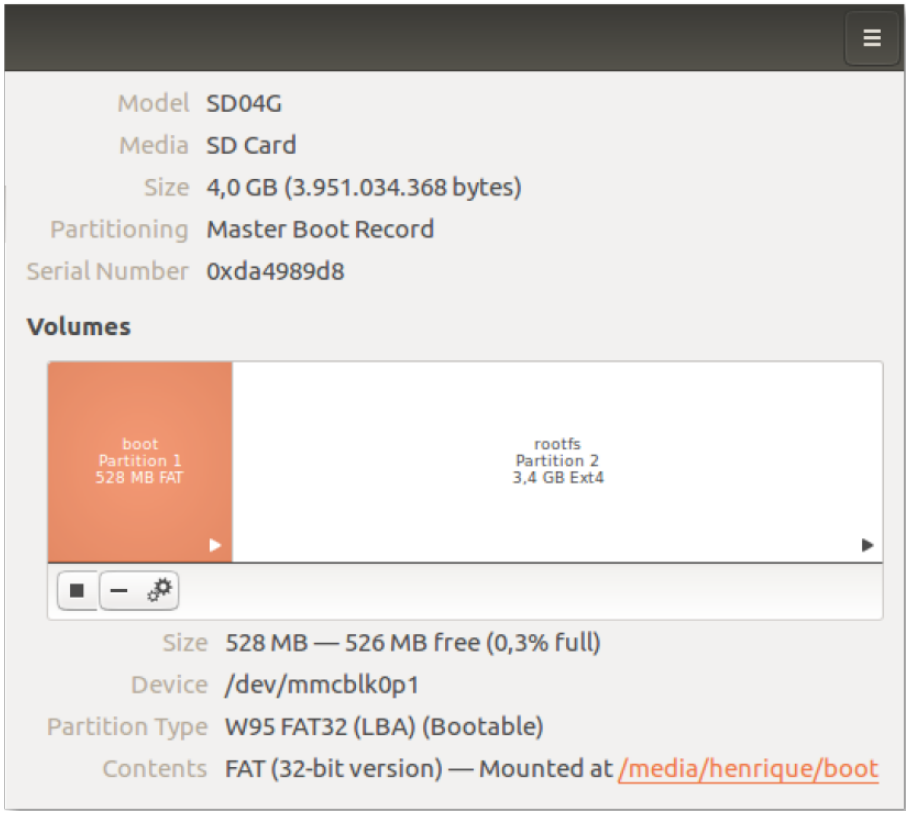
e possibilitar futuras modificações sem grandes dificuldades. Vale ressaltar que este não é o método indicado pelo fabricante, já que os procedimentos recomendados por eles na página Create Bootable MicroSD Card apresentaram os mais diversos erros, todavia, os resultados obtidos utilizando os procedimentos do tópico abaixo são os mesmos.

Particionando o Cartão SD

Este guia descreve o processo de particionamento, utilizando um sistema Linux, de um cartão microSD em duas partes, denominadas de **boot** e **rootfs** com o objetivo de gerar um cartão SD bootável. O procedimento descrito abaixo é realizado utilizando o gerenciador de discos do próprio Ubuntu, não sendo necessário instalar novos *softwares*.

Usualmente, o cartão microSD é configurado em uma única partição formatada no padrão Windows FAT, configuração típica encontrada em cartões adquiridos em varejo. Porém, aqui particionaremos o cartão microSD em duas partes, que serão denominadas **boot** e **rootfs**, sendo o sistema de gestão de arquivos da partição **boot** «VFAT» e da partição **rootfs** «ext4».

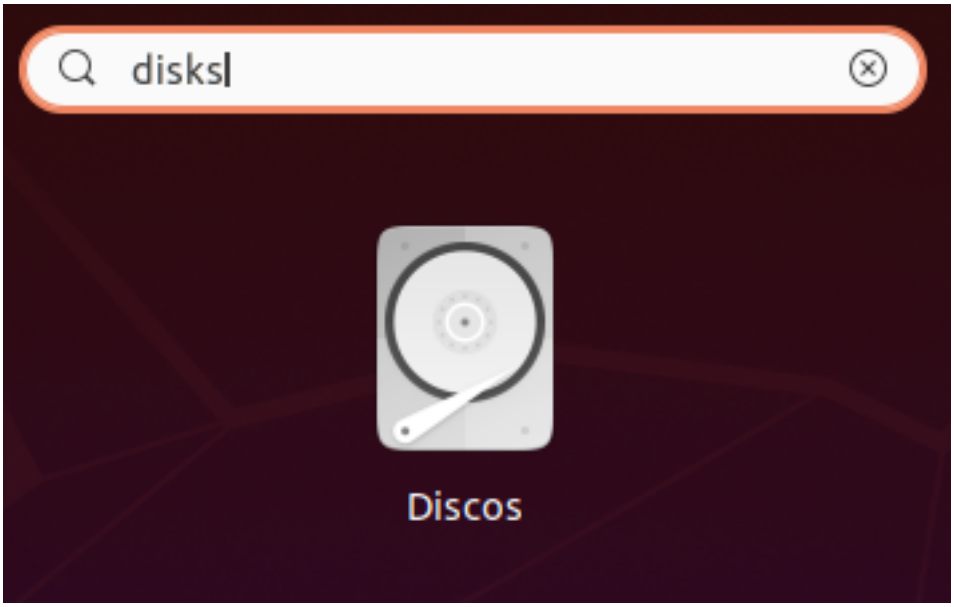
A figura abaixo apresenta um exemplo de cartão de memória com as partições já definidas, montadas e contendo o sistema operacional do computador embarcado. No exemplo o cartão SD possui um total de 4 GB, porém, para o projeto Yocto, um cartão de memória de 2 GB deve ser suficiente.



Procedimentos

> **Aviso:** A versão do sistema operacional utilizada nas atividades foi o Ubuntu 20.04 (LTS), porém os comandos são os mesmos para versões mais antigas do Ubuntu, a partir do Ubuntu 14.04 (LTS). Os procedimentos podem ter algumas diferenças a depender da versão e distribuição do Linux a ser utilizada.

1. Insira o cartão microSD ou um adaptador com ele em uma porta disponível no seu computador Linux.
2. Pesquise em seu computador um aplicativo chamado **Discos** (*Disks*) e o inicie. Logo ao abrir, o aplicativo exibira os dispositivos de memória conectados ao computador.



3. Na aba de **Discos**, selecione o cartão microSD que deseja particionar.

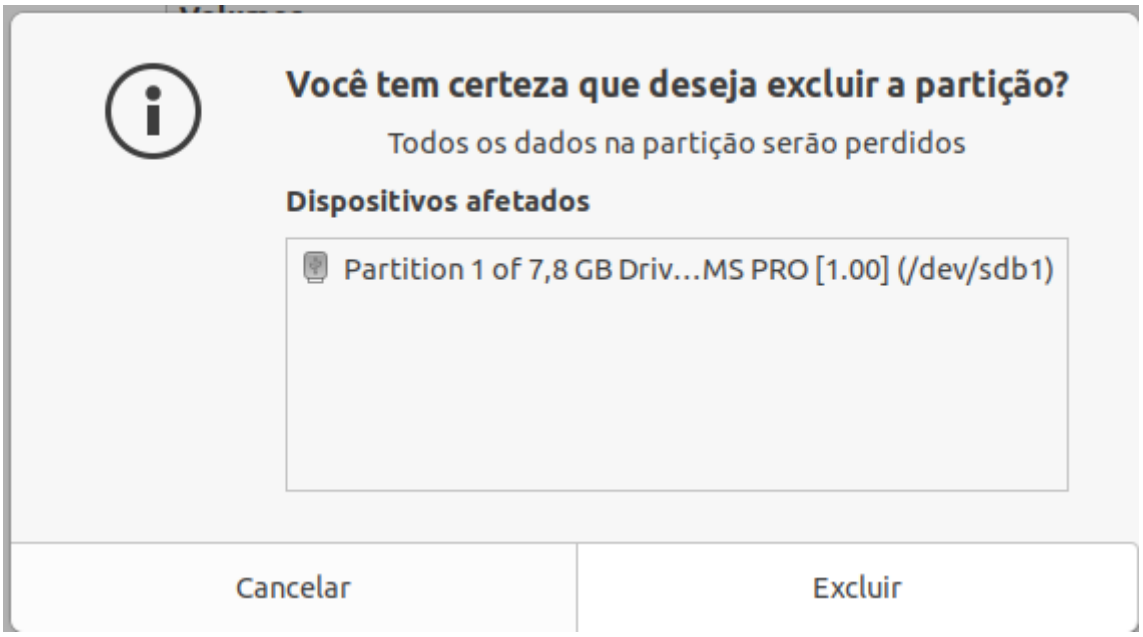
4. Clique em «**Desmontar o sistema de arquivos**» abaixo de **Volumes** para habilitar modificações no cartão microSD.

5. Para criar novas partições em diferentes formatos é recomendado excluir a partição do seu cartão microSD, para isso, clique em «**Excluir partição**».

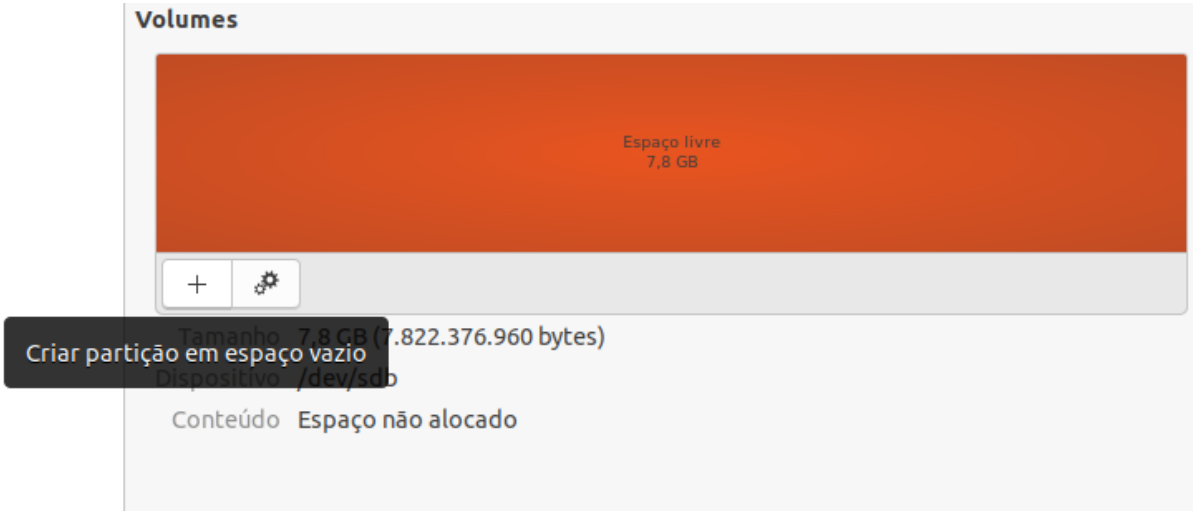

Volumes
SD 8GB
Partição 1
7,8 GB FAT
Desmontar partição selecionada
Tamanho 7,8 GB — 7,8 GB livres (0,2% ocupados)
Dispositivo /dev/sdb1
UUID 51AE-B32C
Tipo de partição W95 FAT32 (LBA)
Conteúdo FAT (32-bit version) — Montado em /media/lucas/SD 8GB


Volumes
SD 8GB
Partição 1
7,8 GB FAT
Excluir partição selecionada
Tamanho 7,8 GB (7.821.328.384 bytes)
Dispositivo /dev/sdb1
UUID 51AE-B32C
Tipo de partição W95 FAT32 (LBA)
Conteúdo FAT (32-bit version) — Não montado

Aviso: Está etapa irá formatar o seu cartão microSD, portanto, todos os dados ali presentes serão excluídos permanentemente.



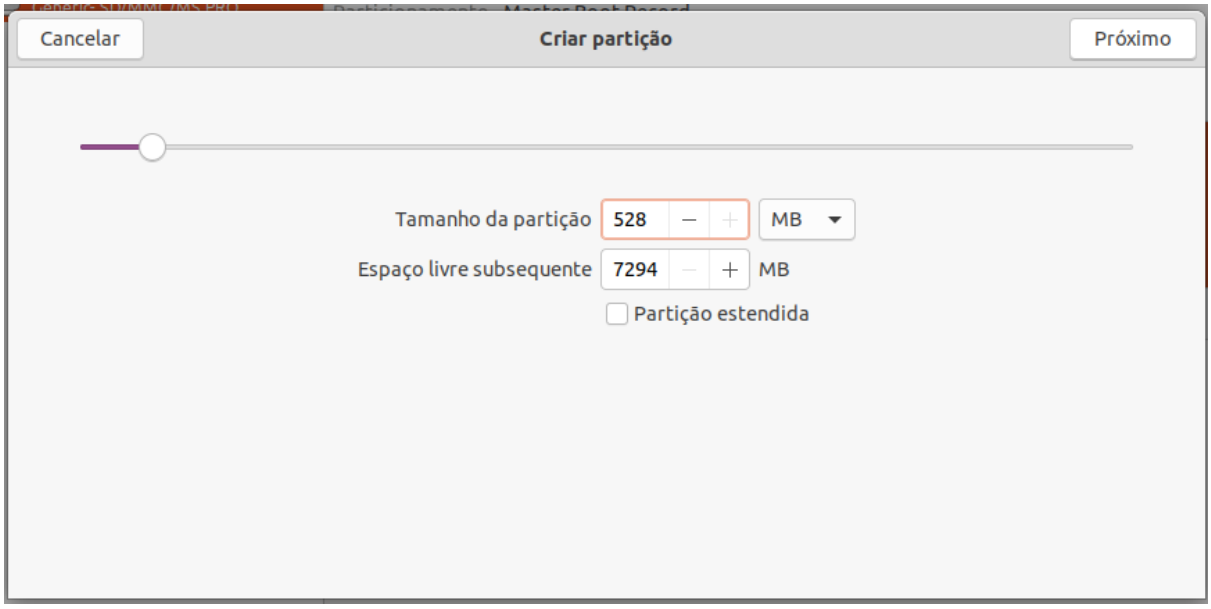
6. Clique em «**Criar uma nova partição** para criar a primeira partição.

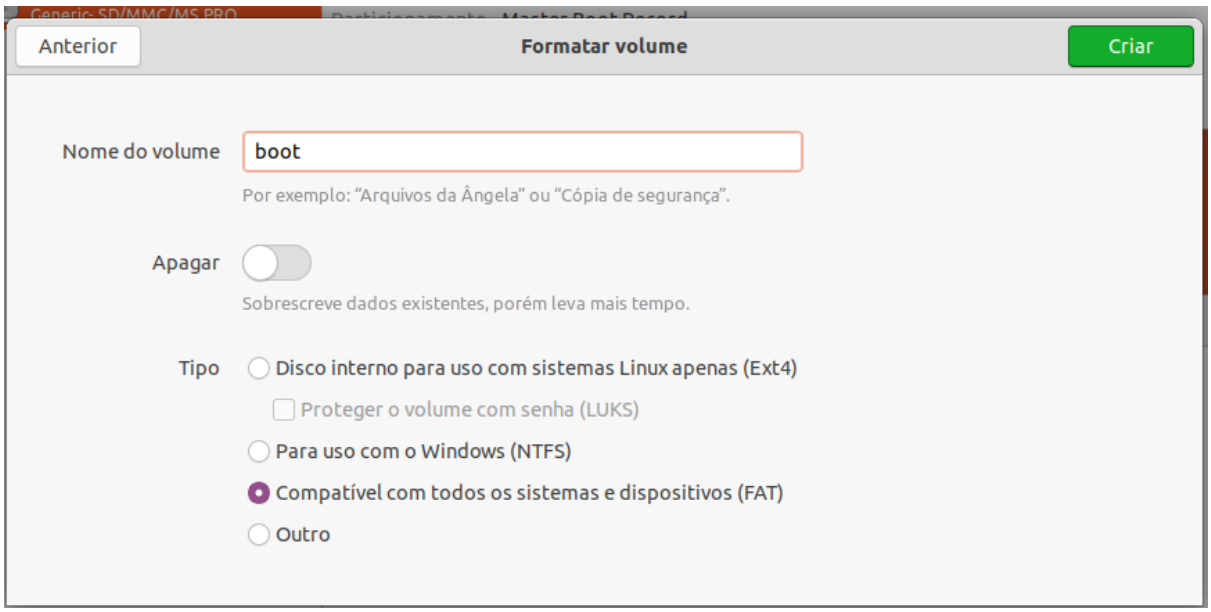


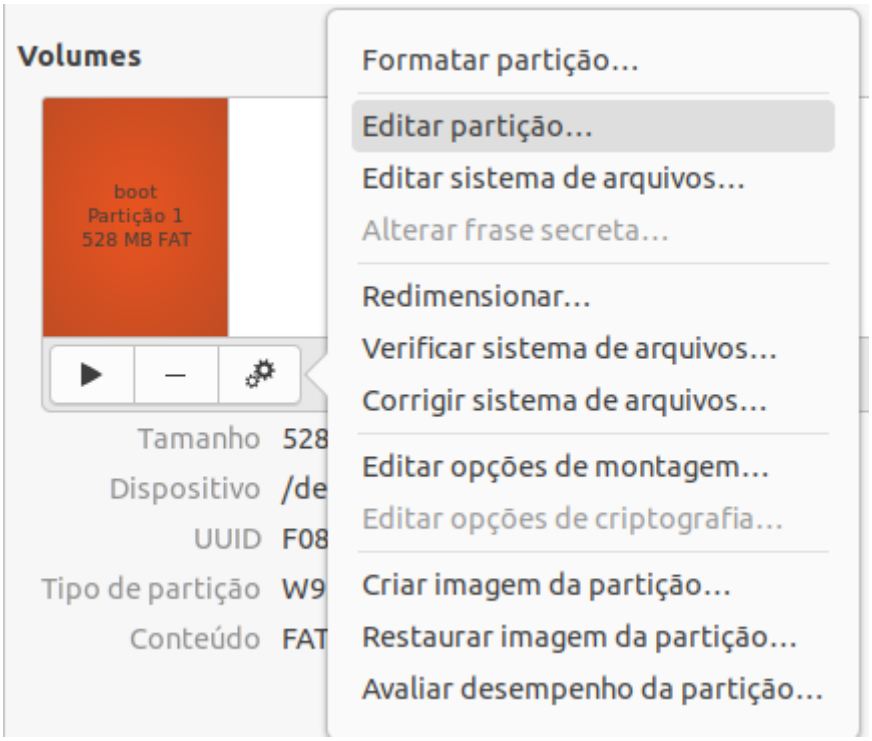
Está partição será nomeada como «**boot**», terá um tamanho de 528MB e será configurado com o tipo de gestão de arquivos «FAT», como exemplificado abaixo. Após configurar, clique em «**Criar**» para gerar está nova partição.

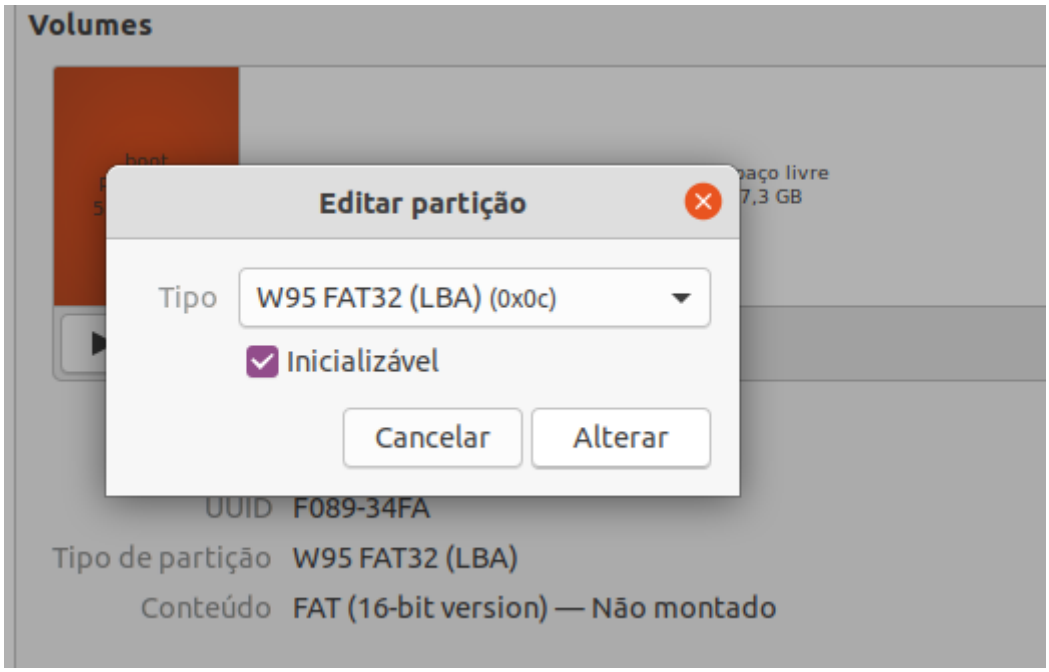
Em seguida, vá em **Mais Ações > Editar partição**, configure o **Tipo de partição** como «**W95 FAT32 (LBA)**» e ative a opção «**Iniciável** para determinar que é nesta partição que o sistema operacional deve ser carregado.

Dica: Neste exemplo, foram reservados 528 MB para a partição de boot, entretanto, utilizam-se para inicialização menos de 100 MB. Sendo assim, caso futuramente venha a faltar espaço para armazenamento de dados será possível ampliar a partição roots refazendo esta divisão.

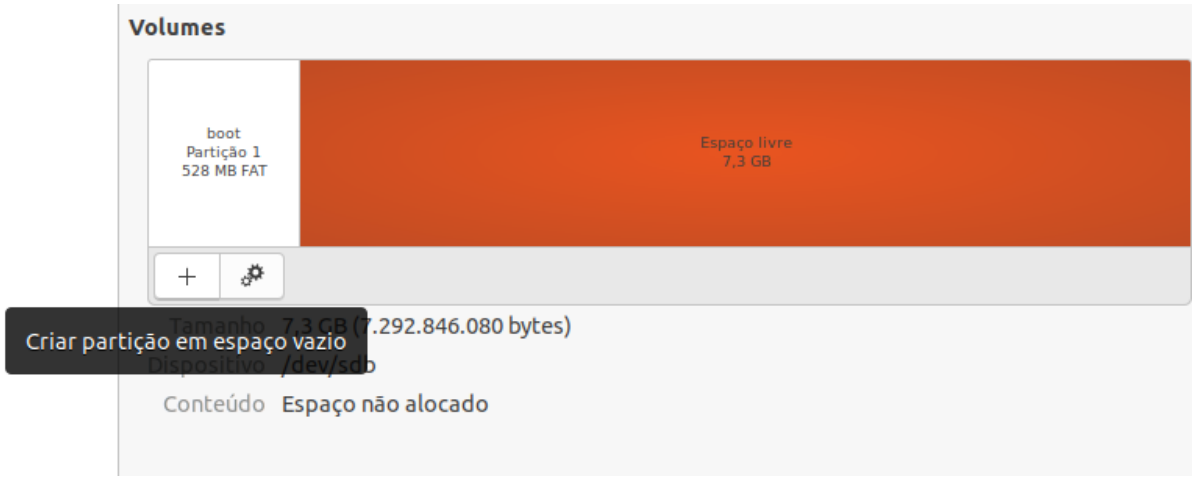

Cancelar
Criar partição
Próximo
Tamanho da partição
528
MB
Espaço livre subsequente
7294
MB
Partição estendida


Anterior
Formatar volume
Criar
Nome do volume
boot
Por exemplo: "Arquivos da Ângela" ou "Cópia de segurança".
Apagar
Sobrescreve dados existentes, porém leva mais tempo.
Tipo
Disco interno para uso com sistemas Linux apenas (Ext4)
Proteger o volume com senha (LUKS)
Para uso com o Windows (NTFS)
Compatível com todos os sistemas e dispositivos (FAT)
Outro


Volumes
boot
Partição 1
528 MB FAT
Tamanho 528
Dispositivo /de
UUID F08
Tipo de partição W9
Conteúdo FAT
Formatar partição...
Editar partição...
Editar sistema de arquivos...
Alterar frase secreta...
Redimensionar...
Verificar sistema de arquivos...
Corrigir sistema de arquivos...
Editar opções de montagem...
Editar opções de criptografia...
Criar imagem da partição...
Restaurar imagem da partição...
Avaliar desempenho da partição...


Volumes
Editar partição
Tipo W95 FAT32 (LBA) (0x0c)
Inicializável
Cancelar
Alterar
UUID F089-34FA
Tipo de partição W95 FAT32 (LBA)
Conteúdo FAT (16-bit version) — Não montado

7. Agora iremos criar a segunda partição, chamada de **rootfs**. Portanto, selecione o espaço livre do cartão SD e clique em **Criar partição em espaço vazio**.



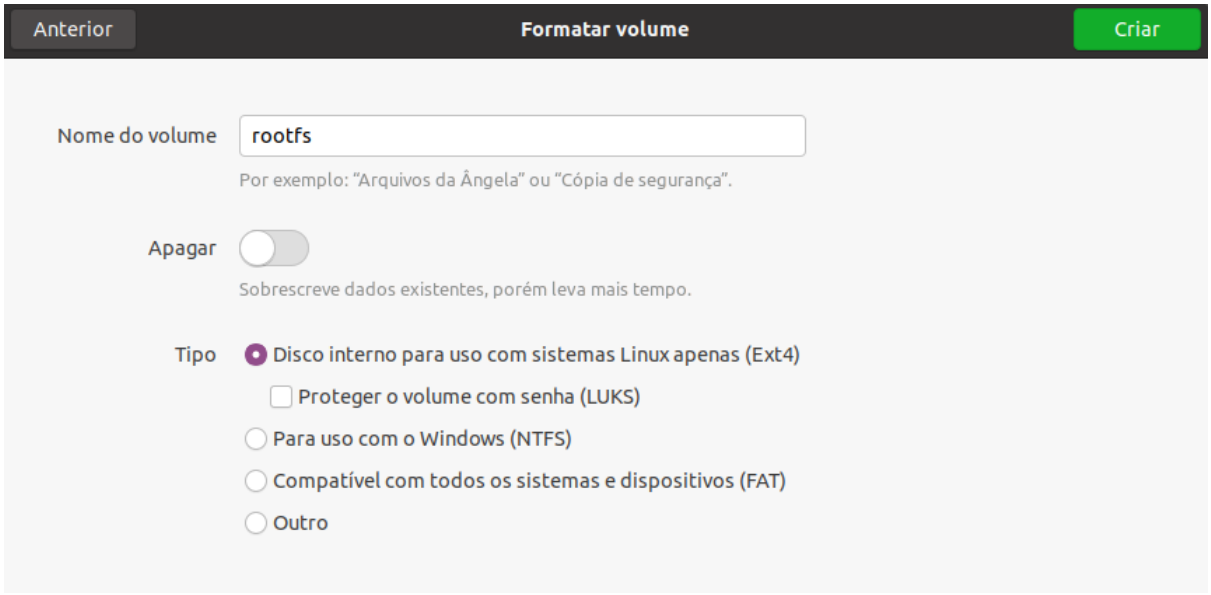
Está partição será nomeada como «**rootfs**» e para ela destinaremos toda a memória remanescente no cartão SD. Esta partição será configurada com o tipo de gestão de arquivos «Ext4», sistema de arquivos padrão dos atuais sistemas GNU/Linux. Após configurar, clique em «**Criar**» para gerar está nova partição.

Em uma execução bem-sucedida, o resultado será similar a figura abaixo, onde os procedimentos foram aplicados em um cartão de 8GB.

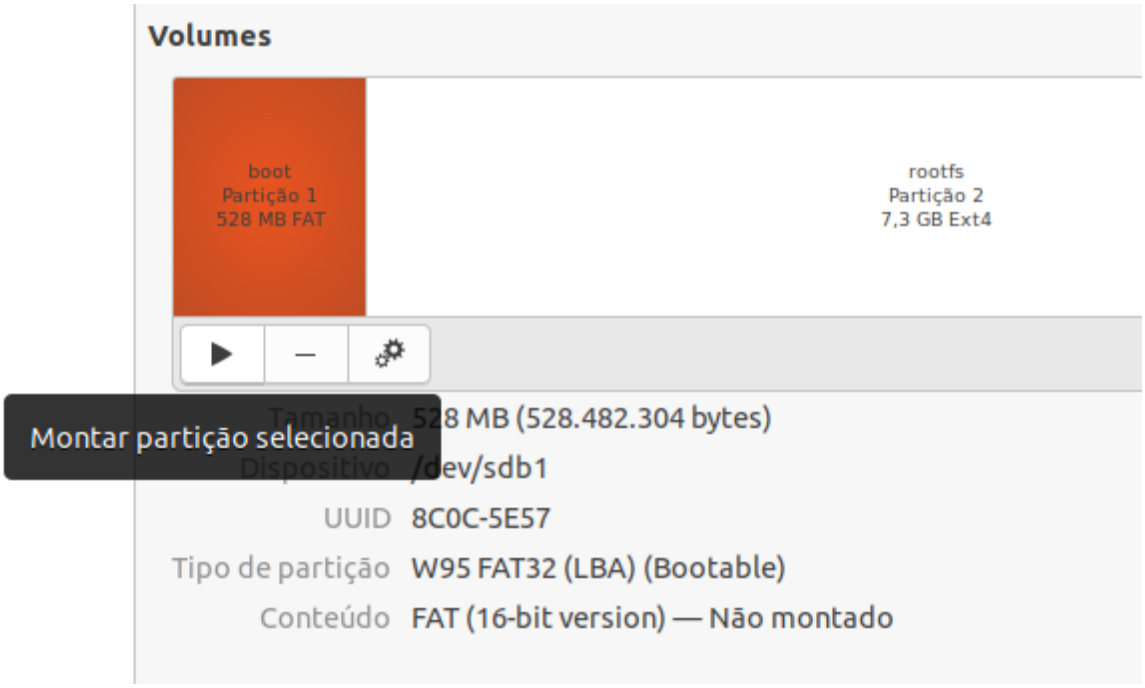
8. (Opcional) Para remontar as partições, basta apenas selecionar a partição e clicar em **Montar a partição selecionada**. Está ferramenta irá montar, automaticamente, a partição selecionada ao sistema de arquivos /media/<User_Name>

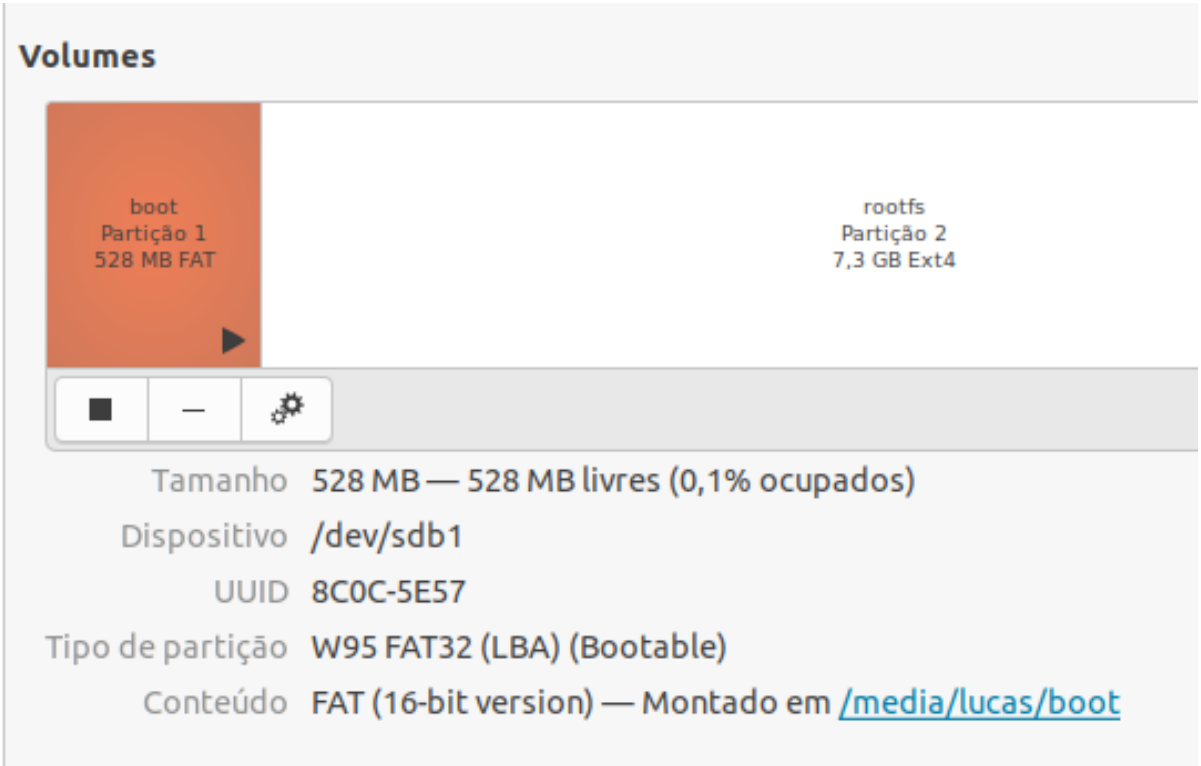
Referências

- Create Bootable MicroSD Card - gumstix.com
- Script - Make 2 Partition SD Card - github.com
- How to Make 2 Partition SD Card - Texas Instruments Processors Wiki


Anterior
Formatar volume
Criar
Nome do volume
rootfs
Por exemplo: "Arquivos da Ângela" ou "Cópia de segurança".
Apagar
Sobrescreve dados existentes, porém leva mais tempo.
Tipo
Disco interno para uso com sistemas Linux apenas (Ext4)
Proteger o volume com senha (LUKS)
Para uso com o Windows (NTFS)
Compatível com todos os sistemas e dispositivos (FAT)
Outro


Modelo Generic- SD/MMC/MS PRO (1.00)
Tamanho 7,8 GB (7.822.376.960 bytes)
Particionamento Master Boot Record
Número de série 20121112761000000
Volumes
boot
Partição 1
528 MB FAT
rootfs
Partição 2
7,3 GB Ext4
Tamanho 528 MB (528.482.304 bytes)
Dispositivo /dev/sdb1
UUID F089-34FA
Tipo de partição W95 FAT32 (LBA) (Bootable)
Conteúdo FAT (16-bit version) — Não montado


Volumes
boot
Partição 1
528 MB FAT
rootfs
Partição 2
7,3 GB Ext4
Montar partição selecionada
Tamanho 528 MB (528.482.304 bytes)
Dispositivo /dev/sdb1
UUID 8C0C-5E57
Tipo de partição W95 FAT32 (LBA) (Bootable)
Conteúdo FAT (16-bit version) — Não montado


Volumes
boot
Partição 1
528 MB FAT
rootfs
Partição 2
7,3 GB Ext4
Tamanho 528 MB — 528 MB livres (0,1% ocupados)
Dispositivo /dev/sdb1
UUID 8C0C-5E57
Tipo de partição W95 FAT32 (LBA) (Bootable)
Conteúdo FAT (16-bit version) — Montado em /media/lucas/boot

- PITA, H. C. Desenvolvimento de sistema de comunicação multiplataforma para veículos aéreos de asa fixa. Faculdade de Tecnologia, Universidade de Brasília, 2018.

Escrevendo a imagem no Cartão MicroSD

Após dividido o cartão SD podemos prosseguir com a instalação do sistema montando suas partições e copiando os arquivos obtidos anteriormente, os dois arquivos bootloaders, para a pasta em que a partição de boot foi montada e extraindo os diretórios do sistema para a pasta em que a partição rootfs foi montada. Depois disso,

O procedimento de montagem de uma partição de memória consiste em uma atividade do sistema operacional para garantir que a transferência de informação será feita da maneira correta, basicamente o dispositivo conectado é lido por inteiro para identificar os arquivos nele armazenados e aonde podem ser escritas novas informações sem que haja sobreposição de dados. Porém mais importante que a montagem da partição é desmontar a partição antes de desconectar o periférico, pois garante que nenhuma atividade de escrita na partição esteja ocorrendo no momento que o dispositivo for removido. Esse procedimento garante, também, que todas as alterações solicitadas tenham sido feitas no periférico e não estejam salvas em arquivos temporários ou buffers do sistema.

O procedimento descrito a seguir é baseado nas recomendações do fabricante e são especificas para a instalação do sistema Yocto Project em dispositivos Gumstix Overo.

Dica: Lembre-se de desmontar as partições antes de remover o cartão SD.

Montagem das Partições

O procedimento de montagem de uma partição de memória consiste em uma atividade do sistema operacional para garantir que a transferência de informação será feita da maneira correta, basicamente o dispositivo conectado é lido por inteiro para identificar os arquivos nele armazenados e aonde podem ser escritas novas informações sem que haja sobreposição de dados.

Para isso, abra o terminal e insira os comandos abaixo:

```
# Comando para montar a partição boot
$ sudo mount t vfat /dev/<mmcblk0p>1/  media/<Nome_de_Usuário>/boot

# Comando para montar a partição rootfs
$ sudo mount -t ext4 /dev/<mmcblk0p>2/  media/<Nome_de_Usuário>/rootfs
```

Nota: Os dois nomes entre <> devem ser mudados. Sobre *Nome_de_Usuário* deve ser mudado para o nome do login da máquina em uso e sobre *mmcblk0p* deve ser mudado para o nome do arquivo que o sistema cria automaticamente quando reconhece o cartão pela primeira vez e esse arquivo estará presente em /dev com nome semelhante ao *mmcblk0p*. Para exibir os dispositivos de mídia conectados ao seu computador, execute o comando `df -hT`.

Dica: **Alternativamente**, é possível realizar o processo de montagem das partições do cartão microSD seguindo o **procedimento 8** do tópico Procedimentos - Particionando o Cartão SD. O resultado esperado é exatamente o mesmo.

Cópia e Extração de Arquivos

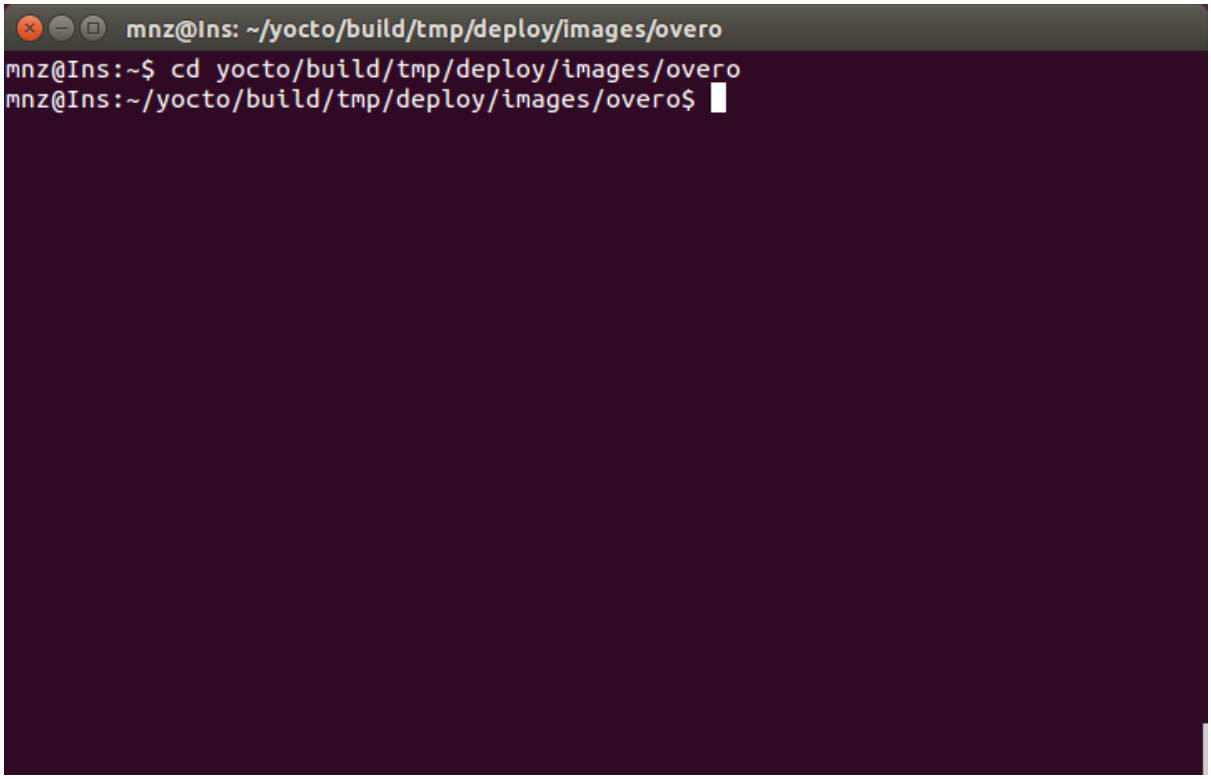
Cópia

Dois arquivos de nome MLO e u-boot.img devem ser copiados para a partição boot do cartão de memória. Para isso, os comandos a seguir devem ser executados dentro da pasta que contém as imagens ou especificando corretamente o local dos arquivos. Recomendamos que entre na pasta das imagens, já que assim é possível realizar a escrita da imagem de forma mais eficiente.

A pasta que contém os arquivos das imagens está localizada em `/yocto/build/tmp/deploy/images/overo` e pode ser acessada pelo comando:

```
$ cd /yocto/build/tmp/deploy/images/overo
```

Após esses comandos, o terminal estará como a imagem abaixo:



Após os passos anterios, digite os seguintes comandos:

```
$ sudo cp MLO /media/<Nome_de_Usuário>/boot/
$ sudo cp u-boot.img /media/<Nome_de_Usuário>/boot
```

Extração

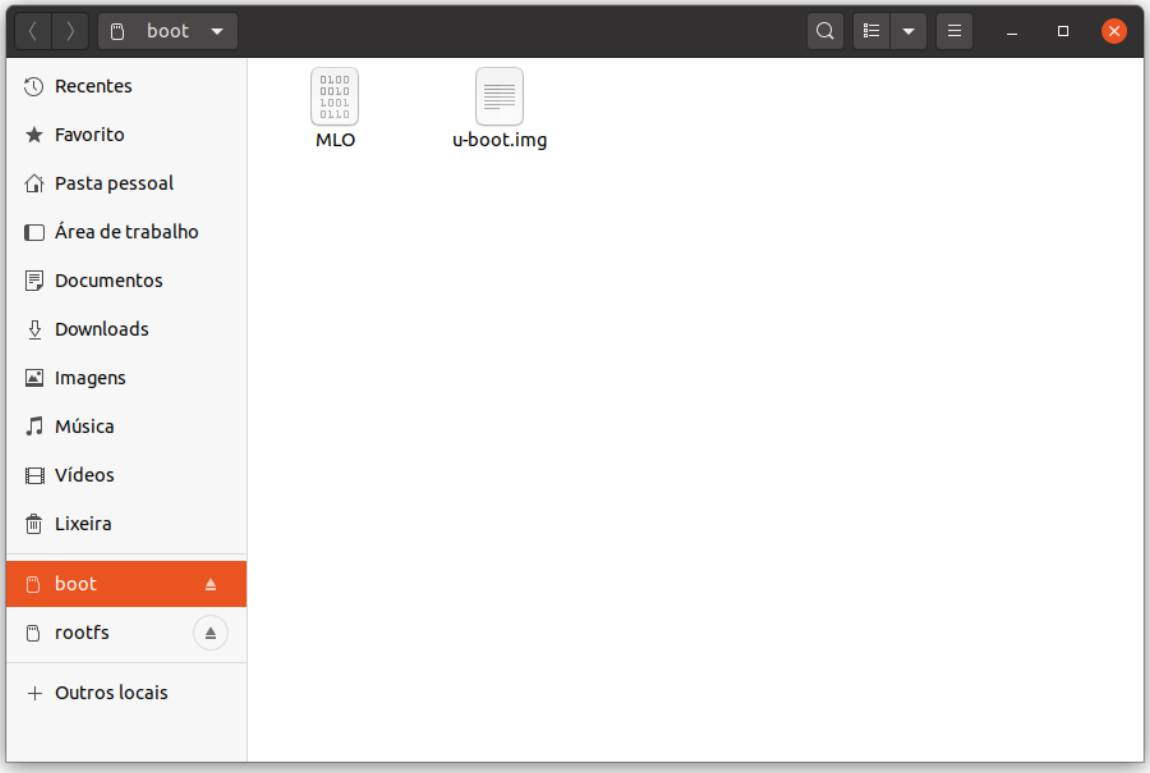
Com o terminal ainda na pasta `/yocto/build/tmp/deploy/images/overo`, a mesma onde foi realizado o procedimento de cópia dos arquivos, insira o seguinte comando para extração do diretório do sistema:

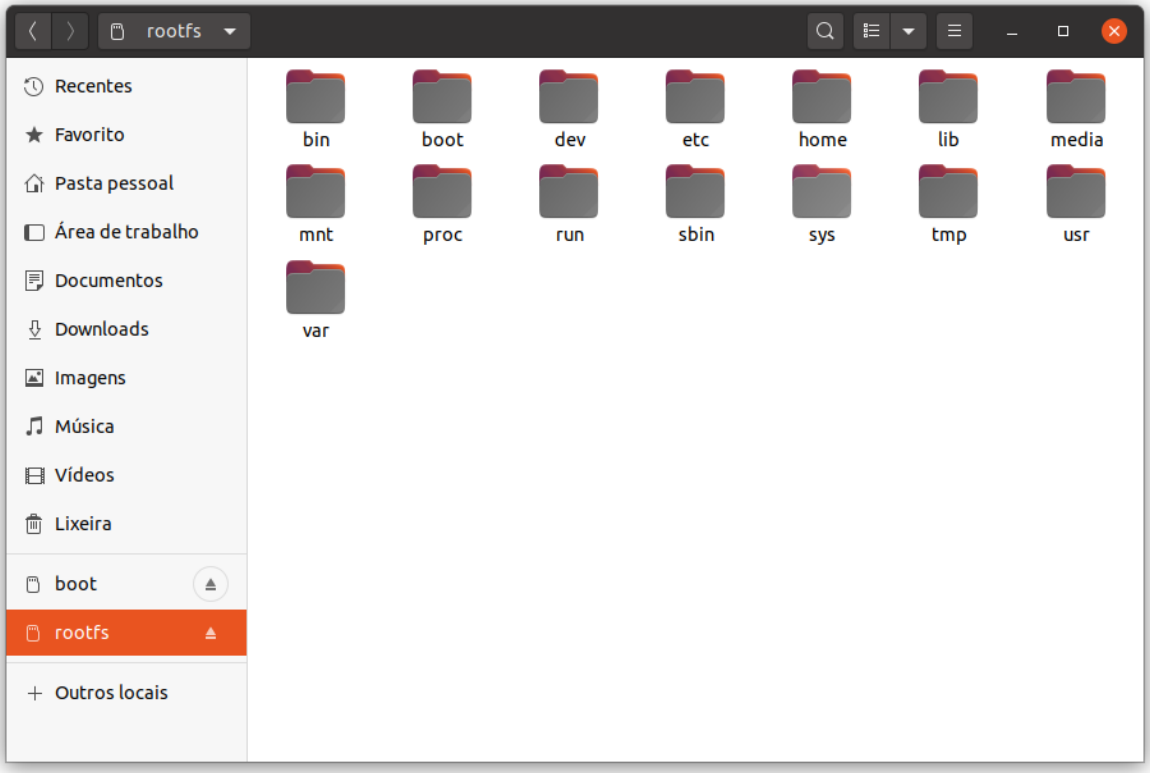
```
$ sudo tar -xjvf gumstix-console-image-overo.tar.bz2 -C /media/<Nome_de_Usuário>/
↪rootfs
```

Resultado no Cartão de Memória

Partição boot:

Partição rootfs:


boot
Recentes
Favorito
Pasta pessoal
Área de trabalho
Documentos
Downloads
Imagens
Música
Vídeos
Lixeira
boot
rootfs
Outros locais
MLO
u-boot.img


rootfs
Recentes
Favorito
Pasta pessoal
Área de trabalho
Documentos
Downloads
Imagens
Música
Vídeos
Lixeira
boot
rootfs
Outros locais
bin
boot
dev
etc
home
lib
media
mnt
proc
run
sbin
sys
tmp
usr
var

Dica: Lembre-se de desmontar as partições antes de remover o cartão SD.

Referências

- Create Bootable MicroSD Card - gumstix.com
- PITA, H. C. Desenvolvimento de sistema de comunicação multiplataforma para veículos aéreos de asa fixa. Faculdade de Tecnologia, Universidade de Brasília, 2018.

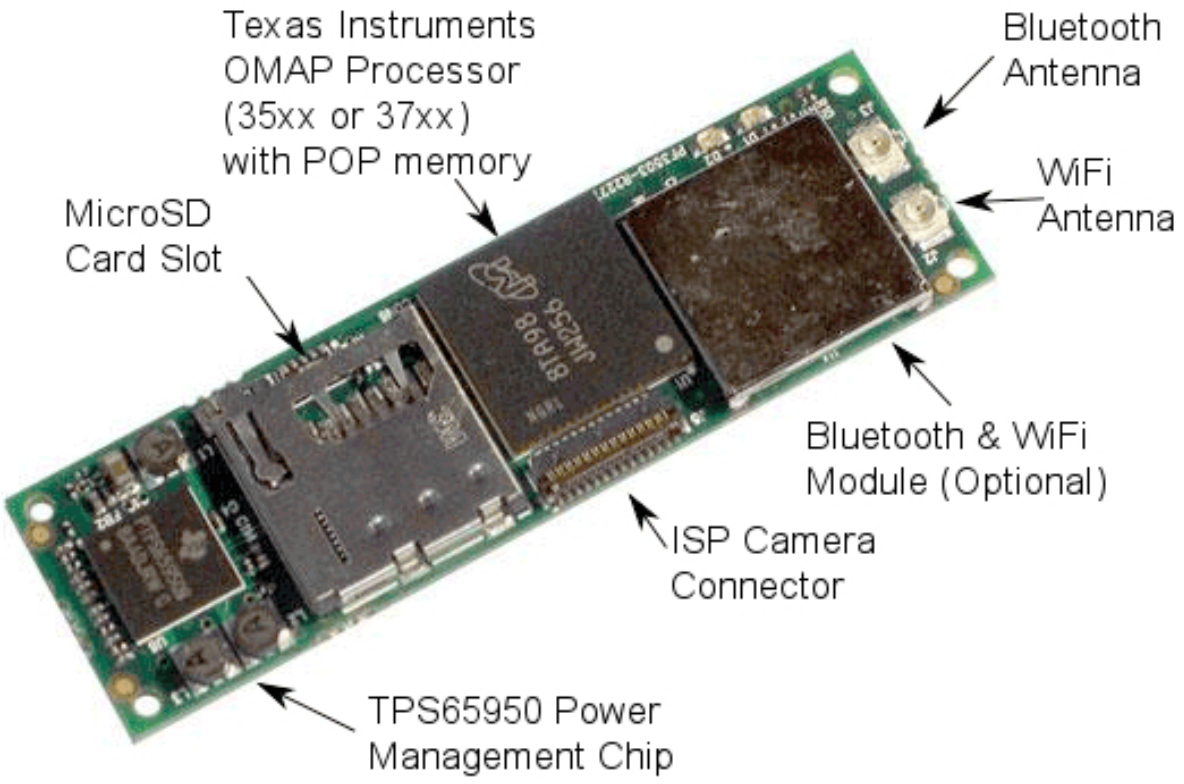
Referências

- PITA, H. C. Desenvolvimento de sistema de comunicação multiplataforma para veículos aéreos de asa fixa. Faculdade de Tecnologia, Universidade de Brasília, 2018.

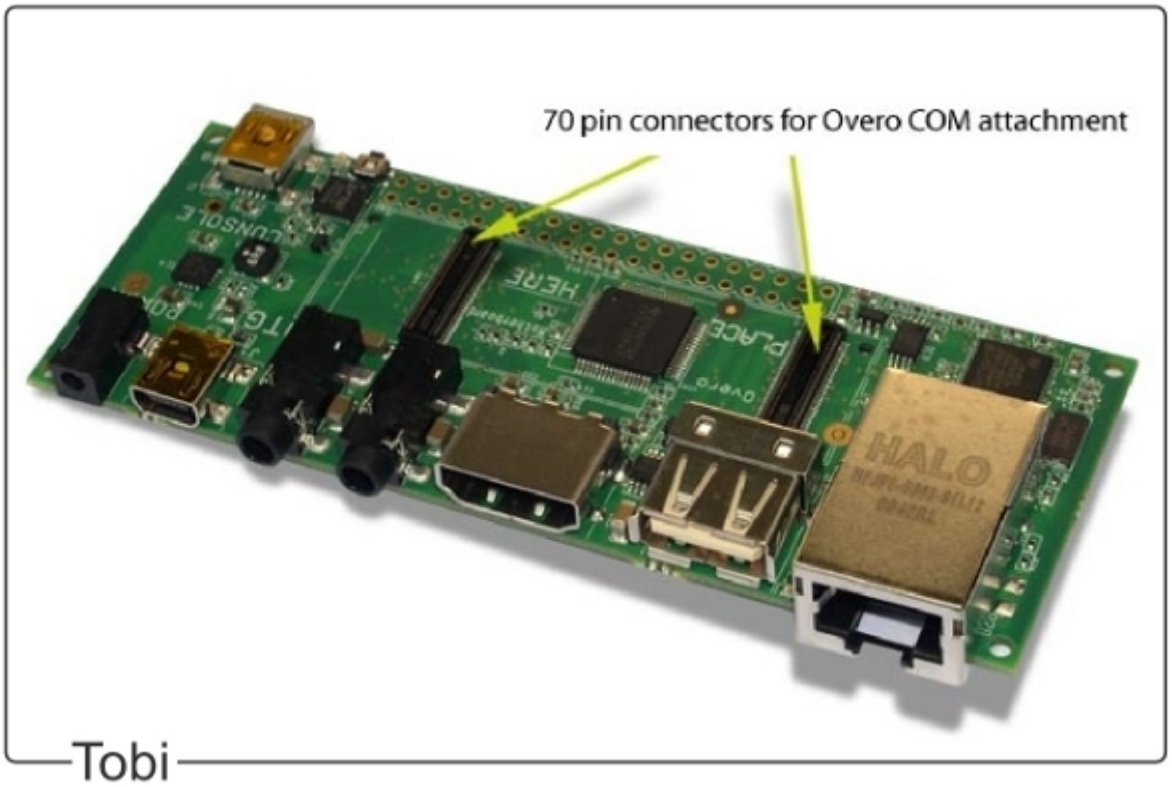
Primeiros passos com o Gumstix Overo

Montando o Gumstix COM na Placa de Expansão Tobi

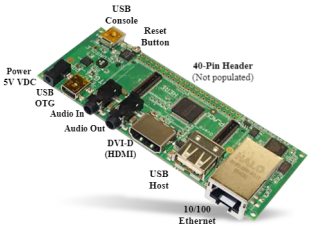
A configuração dos computadores Gumstix Overo consiste em um computador em modulo e uma placa de expansão. O modulo Overo se conecta a uma placa de expansão Tobi através dos dois conectores AVX de 70 pinos localizados na parte inferior do COM. Coloque a placa Tobi em uma superfície plana e antiestética, alinhe a COM com o contorno branco na placa acima dos conectores e pressione delicadamente a COM até que ela se encaixe no lugar.



Para utilizar a câmera, a placa de câmera deve ser conectada à parte superior do Overo COM através de um cabo de fita.



Conexões do Overo



A placa de expansão Tobi vem com uma porta USB Host e uma USB On-the-Go (OTG). A porta USB Host é usada exclusivamente para conectar periféricos ao sistema, enquanto a porta USB OTG pode ser usada para conectar periféricos via cabo USB OTG ou para conectar o sistema Gumstix como periférico a um sistema host separado.

A porta USB Host utiliza uma corrente de 500 mA e aceita uma taxa de sinalização de *High-speed* (HS) a 480 Mbit/s, enquanto a porta USB OTG tem uma corrente de 100 mA e suporta três diferentes taxas de sinalização, *Low Speed* (LS) a 1,5 Mbit/s, *Full Speed* (FS) a 12 Mbit/s e *High Speed* (HS) a 480 Mbit/s.

Nota: Muitos periféricos USB usam uma taxa de sinalização de *Full Speed* (FS) e não funcionam na porta USB Host, que é apenas de *High Speed* (HS). Se você estiver com problemas para conectar periféricos USB diretamente ao sistema Gumstix, conectar os periféricos primeiro a um hub USB com alimentação e depois conectar o hub com alimentação ao sistema Gumstix geralmente resolverá o problema.

Para a conexão de mais periféricos, além da quantidade de portas USB disponíveis na placa de expansão Tobi, recomendamos a utilização de um hub USB. O hub USB **energizado** deve ser conectado a porta USB Host da

placa de expansão e um hub USB **não energizado** deve ser conectado a porta OTG USB da placa de expansão com um cabo USB On-the-Go.

Dica: O vídeo Connecting Gumstix Tobi Expansion Board to Video Monitor demonstra como conectar um Overo COM a um monitor e alguns periféricos através da placa Tobi.

Conectando-se ao Overo

Primeiramente, insira o seu cartão microSD com a imagem do sistema operacional no slot de cartão na parte superior do Overo COM. Certifique-se de que ele se encaixa firmemente no lugar.

O computador Overo pode ser acessado conectando-o a um outro computador Linux ou Windows, ou até mesmo ser ligado diretamente a um monitor DVI e conectado a diversos periféricos, como mouse, teclado, monitor, saída de som, entre outros, através da placa de expansão Tobi.

Neste trabalho, iremos optar por liga-lo a um computador Linux e estabelecer uma conexão seria via a porta USB Console por simplicidade.

Estabelecendo uma conexão serial via console

Para ligar o computador embarcado ao computador conecte um cabo USB ao computador e ao USB console da placa de expansão Tobi. Feito isso, uma luz verde deve se acender indicando a conexão correta. Em seguida verifique em qual porta de comunicação serial Gumstix Overo está conectada, no Windows isso pode ser verificado acessando o **Gerenciador de Dispositivos** e, em seguida, **Portas (COM e LPT)**. No Linux basta executar o comando:

```
$ dmesg | grep tty
```

Nota: O comando `dmesg` é um comando que imprime as mensagens núcleo que, na maioria das vezes, são mensagens dos drivers do dispositivo. Quando acrescentamos `grep tty` estamos realizando uma busca nas saídas da função `dmesg` pelo termo `tty` e restringindo a sua saída aquelas mensagens que contém este termo.

A placa Gumstix deve ser a última entrada a aparecer, por exemplo:

```
user@Ubuntu:~$ dmesg | grep tty
[    0.000000] printk: console [tty0] enabled
[ 4214.120990] usb 2-1: FTDI USB Serial Device converter now attached to␣
↪**ttyUSB0**
```

Em seguida será necessário executar um programa para emular o terminal, recomenda-se o programa Screen para Linux. Caso ainda não o tenha instalado basta executar a linha de comando `sudo apt-get install screen`. Ou no caso do sistema operacional Windows, recomenda-se o PuTTY. Estes programas emulam terminais e executam apenas a tarefa de imprimir os caracteres recebidos pela porta serial, ou USB no caso, e enviar por essa mesma porta os caracteres digitados.

Para iniciar a comunicação por terminal com o Gumstix Overo basta executar a seguinte linha de comando:

```
$ sudo screen /dev/<Nome do dispositivo USB> 115200
```

No caso da linha de comando do exemplo apresentada anteriormente, o termo `ttyUSB0` foi a porta encontrada ao utilizar o comando `dmesg` e «115200» é a velocidade de comunicação em *baud*. Nesse momento a comunicação entre a Gumstix e o computador deve ser estabelecida e assim que a Gumstix for ligada os caracteres devem começar a ser impressos na tela do computador.

Inicializando o Overo COM

Feita conexão com o console, o Overo COM estará pronta para ser ligado. Para inicializar o sistema basta conectar a fonte de alimentação de 5 Volts à sua placa de expansão. Os indicadores LED no COM devem acender em azul e verde. O processo de inicialização será exibido no terminal da sua máquina host.

Porém, antes de ligá-la, é importante comentar que o fabricante recomenda a limpeza de variáveis da memória flash sempre que iniciar uma **nova versão do sistema operacional** no computador embarcado pela primeira vez. Para fazê-lo, basta interromper o processo de boot apertando qualquer tecla antes de seu início quando aparece a mensagem «**Hit any key to stop autoboot**» e uma contagem regressiva na tela. O processo típico de inicialização será similar ao seguinte:

```
reading u-boot.img
reading u-boot.img


U-Boot 2012.04.01 (Jul 19 2012 - 17:31:34)

OMAP36XX/37XX-GP ES1.2, CPU-OPP2, L3-165MHz, Max CPU Clock 1 Ghz
Gumstix Overo board + LPDDR/NAND
I2C:   ready
DRAM:  512 MiB
NAND:  512 MiB
MMC:   OMAP SD/MMC: 0
In:    serial
Out:   serial
Err:   serial
Board revision: 1
Direct connection on mmc2
timed out in wait_for_pin: I2C_STAT=1000
I2C read: I/O error
Unrecognized expansion board
Die ID #2d3800229ff8000001683b060a00b012
Net:   smc911x-0
Hit any key to stop autoboot:  0
Overo #
```

Uma vez interrompido o boot do sistema basta executar o comando `nand erase 240000 20000` para limpar as variáveis salvas e `reset` para reiniciar o processo de boot, como mostrado a seguir:

```
# nand erase 240000 20000
# reset
```

Nota: Se os LEDs azul e verde no COM não acenderem e não for exibido nada no seu terminal, tente pressionar o botão de reset na placa de expansão até ver um processo de inicialização. Se o problema persistir, a imagem pode não ter sido instalada com sucesso. Recomenda-se que você tente instalar novamente ou usar uma imagem diferente.

A figura a seguir ilustra este procedimento. Os caracteres são impressos rapidamente e a contagem de tempo é de apenas 1 segundo para os núcleos do projeto Yocto, portanto é necessário ficar atento para interromper o processo.

Feito isso o processo de boot deve iniciar e diversas mensagens irão aparecer na tela. É importante verificar, na primeira vez que se inicia o sistema operacional, se nenhuma mensagem de erro aparece e, se tudo ocorrer bem, ao final do processo será exigido uma senha, se o computador embarcado chegou a esse ponto provavelmente tudo está em ordem. A senha de acesso ao sistema Yocto é «**root**» e para o sistema Ubuntu gumstix, caso necessário, a senha é igual ao usuário.

Salvando a imagem do SO na memória flash

O computador embarcado Overo WaterSTORM COM da Gumstix conta com uma memória interna não volátil de 1 GB do tipo Flash, memória suficiente para armazenarmos o sistema operacional. Apesar de o mais recomendado ser continuar usando o cartão SD por possuir mais memória e ser transferido entre dispositivos com mais facilidade, ter o sistema operacional salvo na memória flash do computador embarcado pode ser útil.

O site do fabricante descreve quatro maneiras distintas de se realizar este procedimento. A maneira que apresentou o melhor resultado foi a última das opções explicadas e é resumida a instalar na memória flash tudo o que foi instalado no cartão de memória somado ao binário do núcleo através de um script fornecido em seu endereço eletrônico. O script desejado está disponivel em Flashing with U-Boot - Write Images to Flash, porém, todo o processo será detalhadamente descrito a seguir.

1. Com o cartão SD bootavel conectado ao seu computador host, acesse o diretorio **/boot** na partição **rootfs**. Por exemplo, caso o **rootfs** esteja montado em **/media/<Nome_de_Usuário>/rootfs/**:

```
$ cd /media/<Nome_de_Usuário>/rootfs/boot
```

2. Devemos armazenar dentro da pasta **boot** da partição **rootfs** o novo **MLO**, **u-boot.img** e o **binário do núcleo**. Observe que esses *bootloaders* que serão adicionados à pasta **boot** não são os mesmos que estão na partição **boot**, pois estes novos *bootloaders* devem ser específicos para operar da memória flash. Esses novos arquivos podem ser obtidos com os seguintes comandos:

```
$ sudo wget https://s3-us-west-2.amazonaws.com/gumstix-yocto/2015-02-25/overo/
↪master/MLO
$ sudo wget https://s3-us-west-2.amazonaws.com/gumstix-yocto/2015-02-25/overo/
↪master/u-boot.img
$ sudo wget https://s3-us-west-2.amazonaws.com/gumstix-yocto/2015-02-25/overo/
↪master/gumstix-console-image-overo.ubi -O rootfs.ubi
```

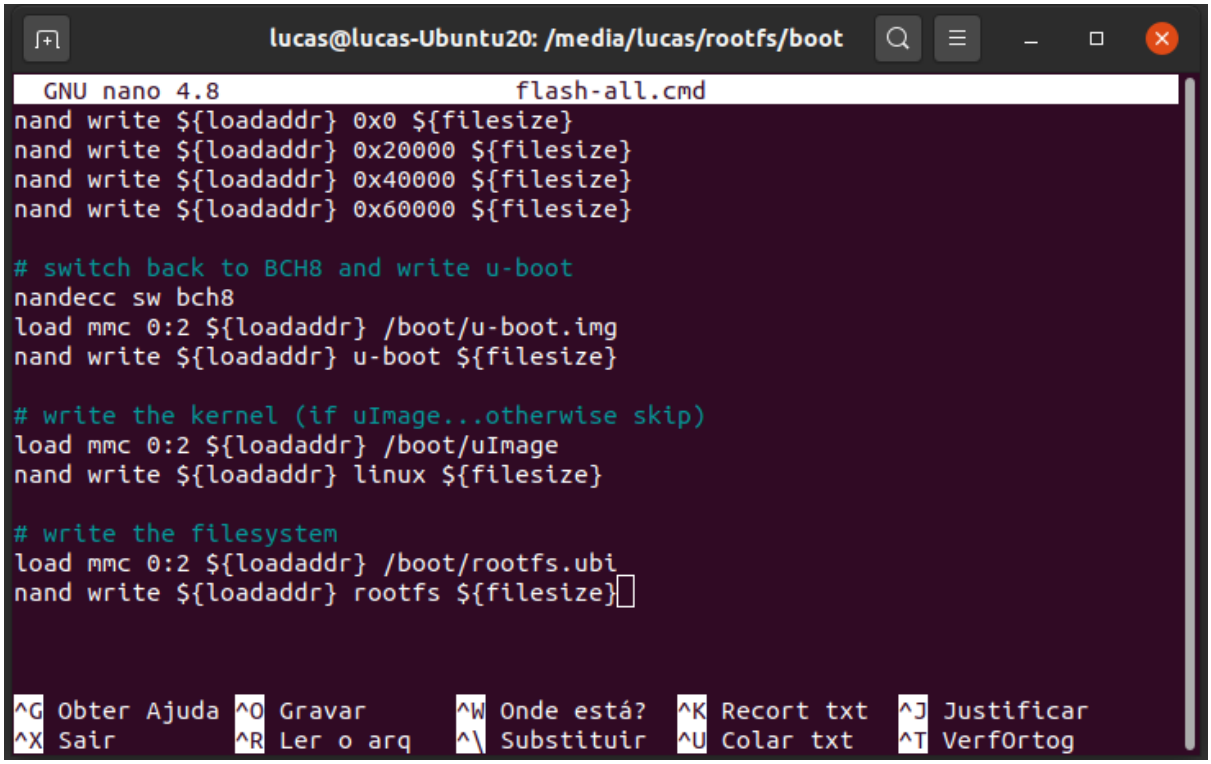
3. Crie um script para gravar os arquivos na memória flash com o nome *flash-all.cmd*. Para isso basta executar o comando:

```
$ sudo nano flash-all.cmd
```

Copiar e colar o script:

```
nand erase.chip

# switch to 1-bit ECC and write MLO
load mmc 0:2 ${loadaddr} /boot/MLO
nandecc hw
nand write ${loadaddr} 0x0 ${filesize}
nand write ${loadaddr} 0x20000 ${filesize}
nand write ${loadaddr} 0x40000 ${filesize}
nand write ${loadaddr} 0x60000 ${filesize}

# switch back to BCH8 and write u-boot
nandecc sw bch8
load mmc 0:2 ${loadaddr} /boot/u-boot.img
nand write ${loadaddr} u-boot ${filesize}

# write the kernel (if uImage...otherwise skip)
load mmc 0:2 ${loadaddr} /boot/uImage
nand write ${loadaddr} linux ${filesize}

# write the filesystem
load mmc 0:2 ${loadaddr} /boot/rootfs.ubi
nand write ${loadaddr} rootfs ${filesize}
```

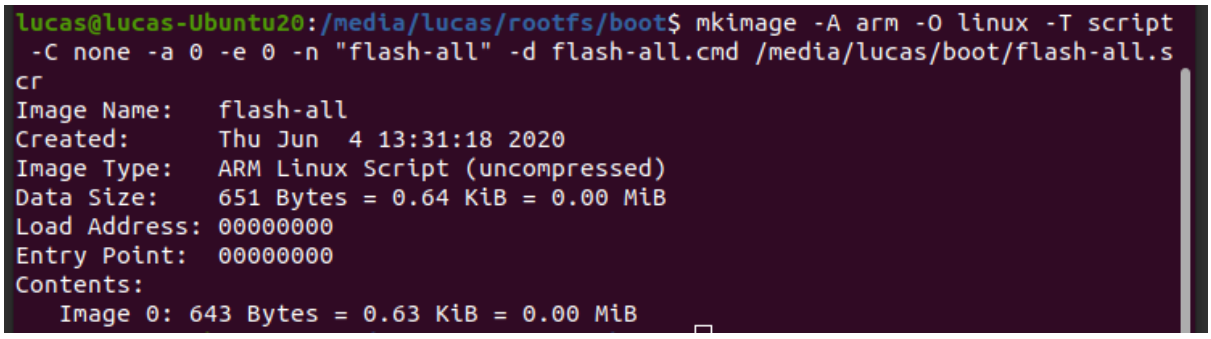
Em seguida confirme o nome do arquivo (**Ctrl+O**) e saia do editor de texto (**Ctrl+X**).



4. Para tornar o script executável e adiciona-lo à partição de boot do cartão SD bootável, basta executar e seguinte linha de comando (assumindo que a partição de inicialização esteja montada em /media/<Nome_de_Usuário>/boot):

Aviso: Lembre-se de editar os nomes dos arquivos no script para coincidirem com os nomes dos arquivos que serão adicionados a seguir.

```
$ mkimage -A arm -O linux -T script -C none -a 0 -e 0 -n "flash-all" -d flash-all.
↪cmd /media/<Nome_de_Usuário>/boot/flash-all.scr
```



Nota: Caso o comando `mkimage` não seja encontrado, basta executar o comando `sudo apt install u-boot-tools` para instalar o pacote de ferramentas em seu computador. O comando `mkimage` é um comando utilizado para fazer imagens para serem utilizadas pelo **u-boot**. As opções de comando e suas explicações são facilmente obtidas digitando `man mkimage` no terminal do Linux.

5. Desmonte o cartão SD e o insira em seu computador embarcado, inicie o sistema e aguarde o carregamento do u-boot. Interrompa o processo de inicialização quando vir «**Hit any key to stop autoboot**» e insira o comando:

```
# mmc rescan 0; load mmc 0 ${loadaddr} flash-all.scr; source ${loadaddr}
```

Essa linha de comando irá executar o script passando os bootloaders, o binário do núcleo e os arquivos raiz do sistema operacional para a memória flash do sistema embarcado e as mensagens apresentadas na figura abaixo devem ser impressas.

```
Overo # mmc rescan 0; load mmc 0 ${loadaddr} flash-all.scr; source ${loadaddr}
mmc - MMC sub system

Usage:
mmc info - display info of the current MMC device
mmc read addr blk# cnt
mmc write addr blk# cnt
mmc erase blk# cnt
mmc rescan
mmc part - lists available partition on current mmc device
mmc dev [dev] [part] - show or set current mmc device [partition]
mmc list - lists available devices
mmc hwpartition [args...] - does hardware partitioning
  arguments (sizes in 512-byte blocks):
    [user [enh start cnt] [wrrel {on|off}]] - sets user data area attributes
    [gp1|gp2|gp3|gp4 cnt [enh] [wrrel {on|off}]] - general purpose partition
    [check|set|complete] - mode, complete set partitioning completed
  WARNING: Partitioning is a write-once setting once it is set to complete.
  Power cycling is required to initialize partitions after set to complete.
mmc setdsr <value> - set DSR register value

reading flash-all.scr
715 bytes read in 5 ms (139.6 KiB/s)
## Executing script at 82000000

NAND erase.chip: device 0 whole chip
Erasing at 0x3ffe0000 -- 100% complete.
OK
57380 bytes read in 117 ms (478.5 KiB/s)

NAND write: device 0 offset 0x0, size 0xe024
 57380 bytes written: OK

NAND write: device 0 offset 0x20000, size 0xe024
 57380 bytes written: OK

NAND write: device 0 offset 0x40000, size 0xe024
 57380 bytes written: OK

NAND write: device 0 offset 0x60000, size 0xe024
 57380 bytes written: OK
397108 bytes read in 142 ms (2.7 MiB/s)

NAND write: device 0 offset 0x80000, size 0x60f34
 397108 bytes written: OK
** File not found /boot/uImage **

NAND write: device 0 offset 0x280000, size 0x60f34
 397108 bytes written: OK
99483648 bytes read in 6315 ms (15 MiB/s)

NAND write: device 0 offset 0xa80000, size 0x5ee0000
corrected bitflip 3249
corrected bitflip 607
 99483648 bytes written: OK
Overo #
```

Retire o cartão SD e reinicie o seu sistema. Se tudo correu bem, seu sistema deve iniciar normalmente.

Referências

- PITA, H. C. Desenvolvimento de sistema de comunicação multiplataforma para veículos aéreos de asa fixa. Faculdade de Tecnologia, Universidade de Brasília, 2018.
- 4. Boot Your System - gumstix.com

- Write Images to Flash - gumstix.com

1.1.4 Utilizando o computador embarcado

Neste capítulo serão discutidos conhecimentos básicos necessários para a manipulação do computador embarcado Gumstix Overo, do sistema operacional linux e do sistema operacional específico instalado e, posteriormente, são realizados testes do GPIO e da comunicação serial do computador embarcado. As informações descritas nessas etapas são essenciais para a compreensão de diversos topicos e serão utilizadas ao longo do trabalho.

Ambientando-se ao Linux

Realizados os procedimentos apresentados na seção anterior de forma correta, o computador embarcado operará com um sistema operacional Linux muito semelhante ao que estamos habituados em computadores regulares. Logo, podemos realizar alguns procedimentos simples para que possamos explorar e nos habituar um pouco o ambiente ao qual vamos trabalhar. O que será demostrado nessa etapa são procedimentos, comandos e informações padrão dos sistemas Linux executados no computador embarcado Gumstix Overo WaterStorm.

Dica: Caso o leitor já esteja habituado ao ambiente de trabalho Linux recomenda-se pular para a próxima seção.

Linhas de Comandos

Começaremos o processo de ambientação apresentando linhas de comando básicas que vão ajuda-lo a utilizar o sistema operacional sem grandes problemas. Vale ressaltar que não há necessidade de nenhum conhecimento especial para utilizar os comandos do Linux, já que o terminal é um programa como qualquer outro.

Dica: Para obter mais detalhes sobre quaisquer comandos listados aqui basta executar o comando seguido de `--help` ou precedido de `man` .

O comando `--help` imprimi uma breve descrição dos comandos seguidos de instruções de uso, por exemplo:

```
uname --help
```

Para reduzir a quantidade de conteúdo impresso pode se usar `less`, por exemplo:

```
ls --help | less
```

Enquanto o comando `man` apresenta o manual de instruções do comando solicitado, por exemplo:

```
man cd
```

Os comandos principais são simples, as setas para **cima e baixo** sobem e descem a página, respectivamente, assim como as teclas de setas para **esquerda e direita** fazem a movimentação para leitura dos textos, o **Enter** também faz a página descer. A tecla «**h**» mostra o *help* do comando `man`, mostrando todas as teclas e atalhos utilizados. E a tecla «**q**» sai da navegação do manual.

---

1. **Comando** `cat`

```
cat [OPÇÃO] [ARQUIVO]
```

Seu nome é uma derivação da palavra *concatenate* (**concatenar**) e permite que você crie, una e exiba arquivos no formato padrão de tela ou em outro arquivo, entre outras coisas.

Se a opção não for especificada, o comando `cat` lê o conteúdo presente no arquivo indicado e o imprime na tela. Por exemplo, ao executar o comando `cat /etc/issue` no terminal do Overo, é impresso o ramo e a versão do sistema operacional utilizado.

```
root@overo:/# cat /etc/issue
Poky (Yocto Project Reference Distro) 1.7.1 \n \l

root@overo:/#
```

Nota: Caso deseje conhecer mais funcionalidades, acesse Comando Cat Linux - hostinger.com.br.

2. **Comando** `uname`

```
uname [OPÇÃO]
```

O comando `uname`, nome derivado do termo «Unix Name», apresenta informações detalhadas sobre o seu sistema Linux, como o nome da máquina, do sistema operacional, do kernel e assim por diante.

Por exemplo, a opção `-a` solicita a impressão de todas as informações disponíveis pelo programa.

```
root@overo:/# uname -a
Linux overo 3.17.8-custom #1 SMP Wed Feb 25 04:05:20 PST 2015 armv7l GNU/Linux
root@overo:/#
```

3. **Comando** `echo`

```
echo [OPÇÃO] [STRING]
```

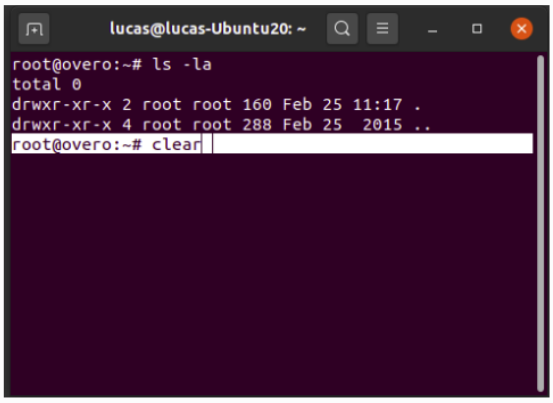
O comando `echo` é um comando utilizado para exibir mensagens na tela ou em um arquivo. Ao utilizar o comando seguido de uma *string*, o texto da *string* é impresso na tela do terminal. Por exemplo:

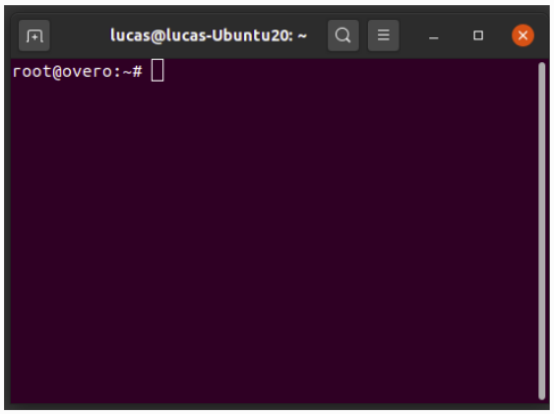
```
root@overo:~# echo "Aerolab"
Aerolab
root@overo:~# echo Aerolab
Aerolab
```

4. **Comando** `clear`

```
clear [OPÇÃO]
```

Utilize o comando `clear` para limpar o conteúdo da tela de seu terminal. O comando não necessita de parâmetros, ele utiliza variáveis do ambiente de trabalho atual para determinar como limpar a tela.





5. **Comando** `pwd`

```
pwd [OPÇÃO]
```

O comando pwd é usado para encontrar o caminho para o diretório atual (da pasta) em que você está. O comando vai retornar um caminho completo, que é basicamente um caminho que começa com uma barra inclinada (/). Por exemplo:

```
root@overo:~# pwd
/home/root
```

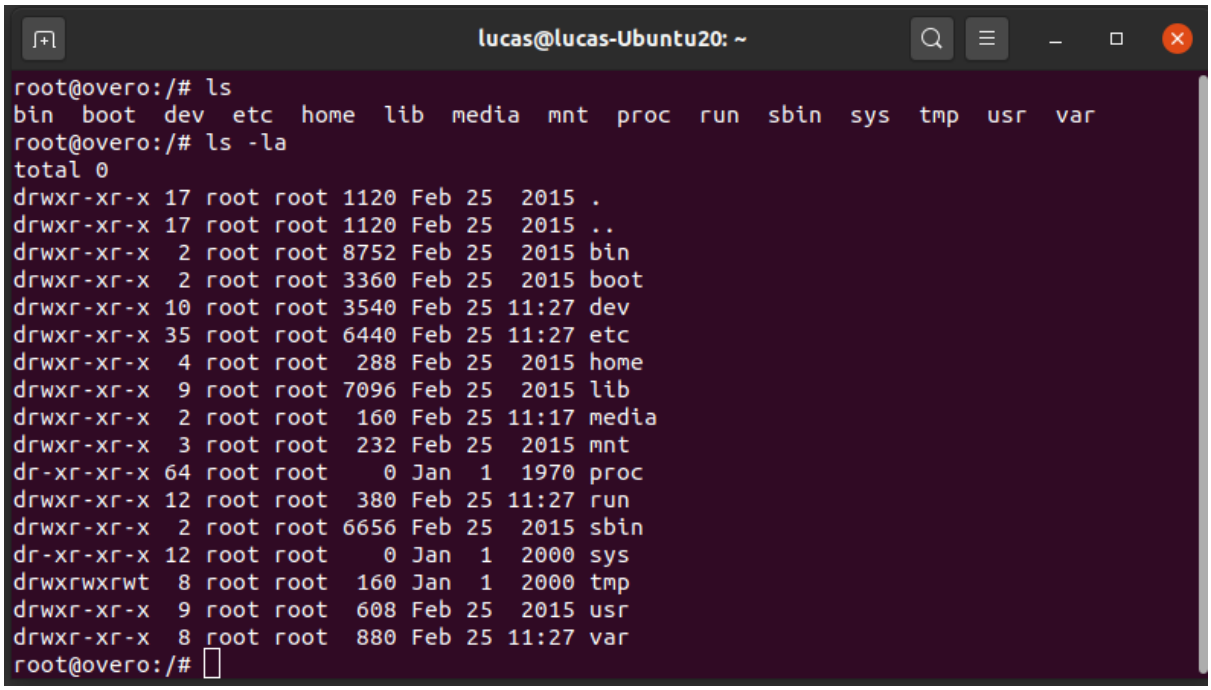
6. **Comando** `ls`

```
ls [OPÇÃO] [ARQUIVO]
```

Seu nome deriva das primeiras consoantes da palavra inglesa *list*. O comando `ls` é usado para listar o conteúdo dentro de um diretório. Por padrão, esse comando vai mostrar apenas os conteúdos do diretório atual em que você estiver.

Ao utilizar o comando `ls` sem especificar nenhuma opção ou diretório, o terminal irá imprimir o conteúdo do diretório atual. Porém, caso deseje ver o conteúdo de outros diretórios, digite **ls**, e então, o caminho do diretório. Por exemplo, escreva `ls /home/username/Documents` para ver os conteúdos de **Documents**.

Uma opção muito util do comando `ls` é a opção `ls -la` que além de listar todos os arquivos e pastas no diretório atual também imprime algumas informações úteis sobre cada um deles.



A figura apresenta um exemplo de saída do comando `ls -la`, nele podemos ver que para cada arquivo é impresso uma linha com várias colunas de informação. Explicar o que cada coluna significa se faz desnecessário, entretanto é importante saber o que as primeiras letras significam, pois muitas vezes essa é a causa de alguns problemas.

As 10 primeiras colunas que são compostas por «**-**» e letras variadas indicam o tipo de arquivo e as permissões dos usuários quanto aqueles arquivos. Na figura, a primeira coluna, que é sempre indicada pela letra «**d**», mostrando que o arquivo é um diretório, se o arquivo fosse um programa ou um arquivo de texto regular este seria indicado por um «**-**«. As noves letras seguintes podem ser separadas em grupos de 3 indicando as permissões do dono, grupo e outros, respectivamente. As letras «**r**», «**w**» e «**x**» indicam **leitura**, **escrita** e **execução**, respectivamente. Se analisarmos, portanto, os dados da pasta «usr» veremos que o dono da pasta possui permissão para ler, escrever e executar, porém seu grupo e outros usuários terão permissão apenas para ler e executar.

7. **Comando** `cd`

```
cd [OPÇÃO] [DIRETÓRIO]
```

Seu nome é um acrônimo da expressão inglesa «*change directory*» (mudar diretório) e sua finalidade é, como sugere seu nome, mudar do diretório atual de trabalho, o diretório em que se está, para um outro diretorio. Por exemplo, caso você esteja em **/home/user** e queira ir para **Documents**, um subdiretório do usuario, basta digitar `cd Documents`.

```
root@overo:/# ls
bin  boot  dev  etc  home  lib  media  mnt  proc  run  sbin  sys  tmp  usr  var
root@overo:/# cd home/
root@overo:/home#
```

Além disso, existem alguns atalhos que podem ser utilizados para navegar rapidamente. São alguns deles:

```
cd ..    # (com dois pontos seguidos) move para um diretório acima (anterior).

cd       # move diretamente para a pasta home.

cd-      # (com um hífen) move para os diretórios anteriores.
```

Nota: O terminal do Linux é sensível a tipos de caracteres. Por isso, você precisa digitar o nome do diretório exatamente como ele é escrito (usando letras minúsculas ou maiúsculas).

8. **Comando** `cp`

```
cp [OPÇÃO] ORIGEM DIRETÓRIO
```

Este comando é usado para copiar arquivos ou diretórios para um diretório específico. Por exemplo, o comando `cp Documento.txt /home/username/Documentos` irá criar uma cópia de **Documento.txt** no diretório **Documentos**, caso este documento exista. Já o comando `cp -R /home/user/projeto /home/user/novo_projeto` irá copiar o diretório **projeto**, com todos seus arquivos, subdiretórios e arquivos dos subdiretórios para o diretório **novo_projeto**.

9. **Comando** `mv`

```
mv [OPÇÃO] ORIGEM DESTINO
```

O nome do comando `mv` deriva das primeiras consoantes da palavra inglesa *move* (mover) e seu uso habitual é mover arquivos, ainda que ele possa também ser usado para renomear arquivos. Ou seja, este comando copia e altera o caminho do arquivo original para o caminho desejado e, desse modo, apaga o arquivo original (sendo possível ainda renomear e mudar o diretório de um arquivo simultaneamente).

A sintaxe neste comando é similar ao comando `cp`. Você precisa digitar `mv`, o nome do arquivo e o diretório de destino. Por exemplo: `mv file.txt /home/username/Documents`.

Já para renomear arquivos, o argumento a ser usado é `mv Nome_Velho.txt Nome_Novo.txt`, sendo «**Nome_Velho.txt**» o arquivo original e «**Nome_Novo.txt**» o novo arquivo.

10. **Comando** `mkdir`

```
mkdir [OPÇÃO] DIRETÓRIO
```

O comando `mkdir` cria um novo diretório, se ele já não existir. Por exemplo, executar `mkdir Test` irá criar um novo diretório chamado **Test**. Seu nome deriva do termo inglês «*Make Directory*», que poderia ser traduzido como «**Criar diretorio**».

11. **Comando** `rmdir`

```
rmdir [OPÇÃO] DIRETÓRIO
```

O comando `rmdir` tem a função de apagar (deletar) um diretório e sua sintaxe é similar à do comando `mkdir`. Porém, este comando só permite que sejam apagados diretórios vazios, sem conteúdo. Seu nome vem do termo em inglês *Remore Directory* (**Remover Diretório**).

12. **Comando** `rm`

```
rm [OPÇÃO] [ARQUIVO]
```

O comando rm é usado para apagar um arquivo específico ou diretório com todos os conteúdos que estiverem lá dentro. Por exemplo, executar o comando `rm /home/user/Documentos/texto.txt` irá apagar o arquivo *texto.txt*.

Caso você deseje deletar um diretório específico (como uma alternativa ao rdmir) use `rm -r [DIRETÓRIO]`.

13. **Comando** `chmod`

```
chmod [OPÇÃO] MODO[,MODO ARQUIVO    #ou
chmod [OPÇÃO] MODO-OCTAL ARQUIVO    #ou
chmod [OPÇÃO] --reference=ARQREF ARQUIVO.
```

O `chmod` (abreviação de *change mode*, em português **alterar modo**) é um comando que pode alterar permissões de acesso de objetos do sistema (arquivos e diretórios) e sinalizações (flags) de modo especial. Os sinalizadores (flags) são uma maneira de definir opções e passar argumentos para os comandos que você executa.

Usualmente, o comando `chmod` é usado na forma:

```
chmod <OPÇÃO> <PERMISSÕES> <NOME DO ARQUIVO>
```

Se nenhuma opção for especificada, o `chmod` modifica as permissões do arquivo para as permissões especificadas. Há duas maneiras de representar as permissões possíveis: com símbolos (caracteres alfanuméricos) ou com números octais (os dígitos de 0 a 7). Aqui iremos nos ater a explicar apenas o método simbólico.

Como visto anteriormente, os caracteres **r, w e x** representam três tipos de permissões: leitura, gravação e execução, respectivamente. Porém, para especificar o grupo de usuarios ao conceder ou remover uma permissão, o comando utiliza mais alguns simbolos. Para visualizar de forma mais clara, imagine que tais símbolos se encontram em duas listas, e a combinação deles gera a permissão:

Grupos de usuários:

```
u: usuário dono do arquivo
g: grupo de usuários do dono do arquivo
O (letra 'o' maiúscula): todos os outros usuários
a: todos os tipos de usuário (dono, grupo e outros)
```

Tipo de permissão:

```
r: se refere às permissões de leitura
w: se refere às permissões de escritura
x: se refere às permissões de execução
```

Para poder combinar os símbolos destas duas listas, usam-se os operadores:

```
+ : adiciona permissão
- : remove permissão
= : define permissão
```

Para mostrar como as combinações podem ser feitas, observe os exemplos abaixo:

```
chmod u+w teste.exe     # adiciona permissão de gravação no arquivo para um usuário

chmod g+rw teste.exe     # adiciona permissão de leitura e gravação ao seu grupo

chmod g=rwx teste.exe     # adiciona todas as permissões disponíveis para o grupo
```

Dica: Como esse comando é relativamente complicado, mais informações podem ser obtidas em Linux chmod Command.

14. **Comando** `sudo`

O comando `sudo` permite que usuários comuns executem tarefas que exigem permissões de outro usuário, em geral o super usuário, para executar tarefas específicas dentro do sistema de maneira segura e controlável pelo administrador. Porém, não é muito aconselhável usá-lo diariamente porque pode ser que um erro aconteça se você fizer algo de errado. O nome é uma forma abreviada de se referir a *Substitute User Do* (**fazer substituição do usuário**) ou *Super User Do* (**fazer como super usuário**).

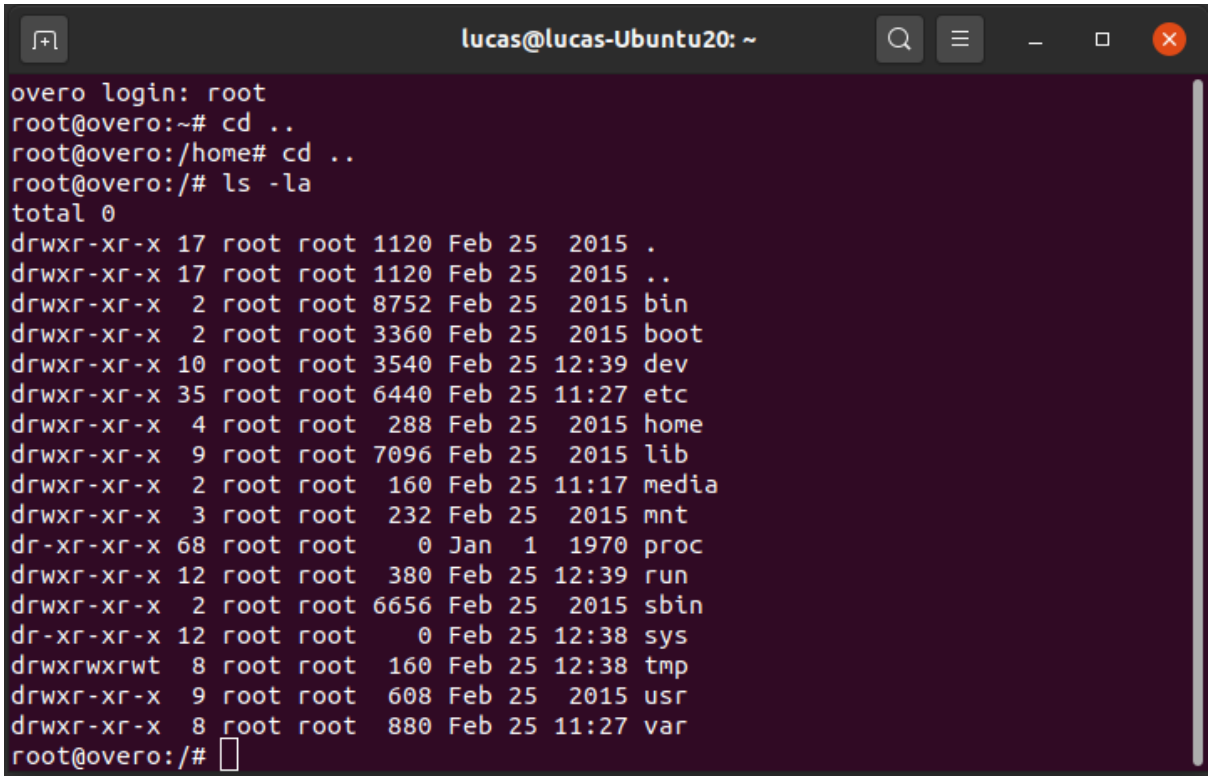
Geralmente, o comando `sudo` é executado na forma:

```
sudo [-u usuário] <comando>
```

Onde <comando> é o comando que deseja executar. A opção [-u usuário] serve para especificar qual usuário deve ser utilizado para executar o comando, se omitida, o comando `sudo` assume o usuário root e pede a senha de login para confirmar.

Explorando os Arquivos do Sistema

Passadas essas informações e estes comandos básicos, já somos capazes de explorar os arquivos do sistema. Portanto, vamos migrar para o primeiro diretório do sistema executando `cd ..` duas vezes. E em seguida executar o comando `ls -la` para que possamos visualizar as pastas do sistema. Se tudo for executado como explicado devemos obter algo como mostrado na figura a seguir.



Dos vários diretórios presentes na figura, destacam-se os diretórios «**/bin**», «**/boot**», «**/dev**», «**/lib**» e «**/sys**».

O diretório «**/bin**» é aonde ficam armazenados os binários dos comandos essenciais do Linux, como os comandos apresentados anteriormente, logo caso se faça necessário acrescentar ao microprocessador mais algum software que se faça necessário ele deve ser adicionado a esta pasta para que possa ser encontrado pelo sistema operacional quando requisitado.

O diretório «**/boot**» já foi utilizado neste trabalho e é o local aonde devem ser armazenados os bootloaders e outros programas que fazem parte da inicialização do sistema.

O diretório «**/dev**» é o diretório onde ficam armazenados os arquivos de dispositivos do sistema. Arquivo de dispositivo é uma maneira que o sistema Linux utiliza para gerar uma interface de comunicação com drivers de dispositivos. Ele será muito utilizado mais para a frente durante a comunicação serial, por exemplo.

O diretório «**/lib**» é o diretório que contém as bibliotecas essenciais para os binários contidos no diretório «/bin», assim caso seja necessário instalação de um novo software provavelmente também precisaremos adicionar alguma biblioteca a este diretório.

Por último, o diretório «**/sys**» é o diretório que contém informações de dispositivos e drivers. Está pasta será muito utilizado caso seja necessário utilizar funções como *general purpose input/output* (**GPIO**), **I2C** e *direct memory access* (**DMA**).

Referências

- PITA, H. C. Desenvolvimento de sistema de comunicação multiplataforma para veículos aéreos de asa fixa. Faculdade de Tecnologia, Universidade de Brasília, 2018.
- Linux man pages online - man7.org
- 30 Comandos Linux Que Todo Usuário Deve Conhecer - hostinger.com.br

Compilação Cruzada

A compilação cruzada ocorre quando um dispositivo compila um código fonte para uma plataforma diferente daquela que compilou o código, geralmente temos um ambiente de desenvolvimento que está em uma arquitetura diferente da arquitetura da placa em que vamos rodar nosso programa. Para que nossos notebooks ou desktops possam gerar um binário, programa que rode em outra arquitetura, acontece o *Cross Compile* (Compilação Cruzada, da tradução direta do inglês). Por exemplo, em nosso caso, um computador com Linux Ubuntu, utilizando um processador Intel ou AMD com arquitetura x86, irá compilar um código para o computador embarcado rodando um sistema Linux adaptado rodando em uma arquitetura completamente diferente.

Utilizar a compilação cruzada para trabalhar com sistemas embarcados é muito comum, principalmente quando estes não possuem capacidade de processamento para suportar um compilador. Para nós seria possível compilar no próprio sistema embarcado, porém, apesar de os recursos não serem tão limitados assim, não queremos desperdiçalos. Além disso programar em um computador regular com a disposição de diversos tipos de IDE diferentes é muito melhor do que programar em um computador embarcado com recursos de interface limitadas.

Para realizarmos esse processo de compilação cruzada precisamos primeiro obter um *software development kit*, também conhecido como Kit de desenvolvimento de software, SDK ou *devkit*. O SDK nada mais é que um conjunto de ferramentas de desenvolvimento de software que possibilita a criação e compilação dos softwares para um sistema diferente.

Instalando o SDK

Nota: O projeto Yocto oferece um tutorial de como obter e instalar o SDK para seu sistema na página do GitHub Cross Compile with Yocto SDK. Todavia, todos os procedimentos realizados e os resultados obtidos na instalação serão descritos nesse tópico.

Para obter o SDK para a imagem do sistema operacional que estamos utilizando basta executar os procedimentos a seguir:

1. **Criação dos diretórios**

Criaremos um diretório chamado **workspace** onde você não apenas instalará o SDK, mas também compilará o código de exemplo. Escolha um local conveniente para você e execute os comandos.

```
$ mkdir workspace
$ cd workspace
$ wget http://gumstix-yocto.s3.amazonaws.com/sdk.sh
```

```
lucas@lucas-Ubuntu20:~$ mkdir workspace
lucas@lucas-Ubuntu20:~$ cd workspace/
lucas@lucas-Ubuntu20:~/workspace$ wget http://gumstix-yocto.s3.amazonaws.com/sdk
.sh
--2020-06-15 12:24:24--  http://gumstix-yocto.s3.amazonaws.com/sdk.sh
Resolvendo gumstix-yocto.s3.amazonaws.com (gumstix-yocto.s3.amazonaws.com)... 52
.218.245.106
Conectando-se a gumstix-yocto.s3.amazonaws.com (gumstix-yocto.s3.amazonaws.com)|
52.218.245.106|:80... conectado.
A requisição HTTP foi enviada, aguardando resposta... 200 OK
Tamanho: 1028162585 (981M) [text/x-shellscript]
Salvando em: “sdk.sh”

sdk.sh                      0%[                                    ]   3,13M   41
sdk.sh                0%[                    ]   7,41M   343KB/s    TED 47m 32s
```

Nota: O comando `wget` irá efetuar download do SDK para sistemas Gumstix. Esse processo pode demorar um pouco, a depender da sua velocidade de download.

2. **Verificando as permissões**

Após o fim do download, verifique as permissões do arquivo executando o comando `ls -la sdk.sh`. Se o terminal retornar que há permissão para executar este arquivo, pule para o próximo procedimento. Caso contrário, será necessário alterar as permissões do arquivo através do comando `chmod`.

Nota: Relembrando, os três primeiras espaços indicam as permissões do dono e, basicamente, temos 3 permissões básicas, «r», «w» e «x» indicando permissão para leitura, escrita e execução, respectivamente.

Por exemplo, na imagem abaixo, o arquivo possui permissão para leitura e escrita, mas não pode ser executado.

```
lucas@lucas-Ubuntu20:~/workspace$ ./sdk.sh
bash: ./sdk.sh: Permissão negada
lucas@lucas-Ubuntu20:~/workspace$ ls -la sdk.sh
-rw-rw-r-- 1 lucas lucas 1028162585 mai 23 05:35 sdk.sh
lucas@lucas-Ubuntu20:~/workspace$
```

Para adicionar a permissão para execução, basta executar o comando `chmod +x sdk.sh`.

```
lucas@lucas-Ubuntu20:~/workspace$ chmod +x sdk.sh
lucas@lucas-Ubuntu20:~/workspace$ ls -la sdk.sh
-rwxrwxr-x 1 lucas lucas 1028162585 mai 23 05:35 sdk.sh
lucas@lucas-Ubuntu20:~/workspace$
```

3. **Instalação**

Para iniciar a instalação, execute o arquivo *sdk.sh*. Assim que solicitado, digite o diretório que deseja instalar o SDK (recomendamos a criação de um repositório chamado *sdk*) e confirme.

```
$ ./sdk.sh
```

4. **Configurando o ambiente**

```
lucas@lucas-Ubuntu20:~/workspace$ ./sdk.sh
Poky (Yocto Project Reference Distro) SDK installer version 2.7.3
==================================================================
Enter target directory for SDK (default: /opt/poky/2.7.3): sdk
You are about to install the SDK to "/home/lucas/workspace/sdk". Proceed [Y/n]?
y
Extracting SDK......
```

```
lucas@lucas-Ubuntu20:~/workspace$ ls
sdk  sdk.sh
lucas@lucas-Ubuntu20:~/workspace$ cd sdk/
lucas@lucas-Ubuntu20:~/workspace/sdk$ ls
environment-setup-cortexa8hf-neon-poky-linux-gnueabi
site-config-cortexa8hf-neon-poky-linux-gnueabi
sysroots
version-cortexa8hf-neon-poky-linux-gnueabi
lucas@lucas-Ubuntu20:~/workspace/sdk$
```

A instalação do SDK irá gerar uma pasta com o conteúdo apresentado na figura abaixo.

Nota: O diretório «sysroot» contém os arquivos raiz dos dois sistemas, tanto do sistema que irá compilar o código quanto do sistema que irá executar o programa, e o primeiro arquivo da lista importa os endereços e variáveis importantes para a compilação do código.

Para podermos prosseguir com a compilação do código é necessária a execução da seguinte linha de comando:

```
source sdk/environment-setup-cortexa8hf-neon-poky-linux-gnueabi
```

```
lucas@lucas-Ubuntu20:~/workspace$ source sdk/environment-setup-cortexa8hf-neon-p
oky-linux-gnueabi
lucas@lucas-Ubuntu20:~/workspace$
```

Compilando Hello World

Uma vez realizada a instalação, podemos testar a compilação cruzada criando um simples script. Vamos criar um arquivo no editor de texto chamado *helloworld.c*, colar o código abaixo e salvar no diretório *workspace*.

```
#include <stdio.h>
int main(void)
{
printf ("Hello World!\n");
return 0;
}
```

Agora podemos executar o comando `make <nome_do_código>` para criar um arquivo binário executável do *helloworld.c*.

Nota: O comando «make» é na verdade a simplificação de uma extensa linha de comando que chama um compilador **arm-poky-Linux-gnueabi-gcc** e dá a ele os parâmetros contidos na pasta SDK. Tudo isso graças ao comando «source» utilizado anteriormente.

Uma vez obtido o executável do código basta copiá-lo para uma das pastas do cartão de memória transferi-lo para o Overo e executá-lo. Lembre-se que o diretório principal é o diretório `/home/root/`, então se o arquivo for colocado dentro deste diretório será bem fácil encontra-lo.

```
lucas@lucas-Ubuntu20:~/workspace$ ls
helloworld.c  sdk  sdk.sh
lucas@lucas-Ubuntu20:~/workspace$ make helloworld
arm-poky-linux-gnueabi-gcc  -mfpu=neon -mfloat-abi=hard -mcpu=cortex-a8 --sysroo
t=/home/lucas/workspace/sdk/sysroots/cortexa8hf-neon-poky-linux-gnueabi  -O2 -pi
pe -g -feliminate-unused-debug-types   -Wl,-O1 -Wl,--hash-style=gnu -Wl,--as-nee
ded  helloworld.c   -o helloworld
lucas@lucas-Ubuntu20:~/workspace$
```

Depois de inserido o cartão de memória no Overo, podemos inicia-lo normalmente. Quando iniciado, vamos até o diretório em que o programa foi salvo e o executamos com o comando `./nome_do_código`. Se tudo ocorrer bem, o programa deverá ser executado, similar a figura abaixo.

```
Poky (Yocto Project Reference Distro) 1.8.2 overo ttyO2

overo login: root
root@overo:~# ls
helloworld
root@overo:~# ./helloworld
Hello World!
root@overo:~#
```

Referências

- PITA, H. C. Desenvolvimento de sistema de comunicação multiplataforma para veículos aéreos de asa fixa. Faculdade de Tecnologia, Universidade de Brasília, 2018.
- Cross Compile with Yocto SDK - github.com/gumstix

Registradores

Aviso: Este tópico precisa ser testado e reeditado.

Seguindo os procedimentos das seções anteriores somos capazes de iniciar o sistema e gerar programas a serem executados pelo sistema operacional. O próximo passo é, portanto, controlar os sinais que podem ser enviados a outros dispositivos pelo computador embarcado para estabelecer a comunicação entre os dispositivos.

A comunicação entre dispositivos é feita pela alteração dos níveis de tensão dos pinos do computador embarcado. Esses pinos estão, de uma maneira resumida, conectados a **espaços de memória do sistema** e, quando alteramos o bit armazenado neste espaço de memória, alteramos também o nível de tensão do pino, permitindo a codificação de uma mensagem e sua transmissão a outro dispositivo.

Posteriormente, a comunicação entre dispositivos será mais discutida, mas neste momento o que mais nos importa são os **espaços de memória**, citados no parágrafo anterior. Esses espaços de memória são na verdade circuitos digitais voláteis que são capazes de armazenar níveis de tensão, o acesso ao conteúdo desses espaços de memória é extremamente rápido e a estes espaços de memória é dado o nome de **registrador**. Os registradores estão no topo da hierarquia de memória, sendo assim o tipo de memória mais rápida de uma unidade central de processamento.

Dessa forma, para que possamos implementar a comunicação entre dois dispositivos, um modem e o computador embarcado por exemplo, precisamos, primeiro, executar uma tarefa mais simples de alterar os níveis de tensão de um pino. Esse processo de alterar os níveis de tensão de um pino possui diversas aplicações que vão desde o simples controle de **ON/OFF** de um LED até comunicação serial entre dispositivos. Aos pinos com esse propósito é dado o nome de General Purpose Input/Output (**GPIO**).

General Purpose Input/Output (**GPIO**) são, basicamente, pinos de comunicação de entrada e saída de sinais digitais, de um circuito integrado ou placa de circuito eletrônico, sem finalidade pré-definida, podendo assim ter funções definidas pelo projetista ou usuário para prover uma interface entre outros dispositivos (periféricos, modens, microcontroladores, microprocessadores etc.).

Como comentado anteriormente, estamos utilizando o computador embarcado Overo junto a uma placa de expansão Tobi. Uma das funções desta placa é fornecer acesso ao usuário aos pinos do computador embarcado, portanto os pinos do computador embarcado que podemos acessar fisicamente são os pinos da placa de expansão Tobi. Na figura abaixo podemos visualizar um diagrama que contém, de maneira resumida, quais funções ou pinos do computador embarcado estão conectadas a cada pino da placa de expansão Tobi. Observe que alguns desses pinos possuem mais de uma função.

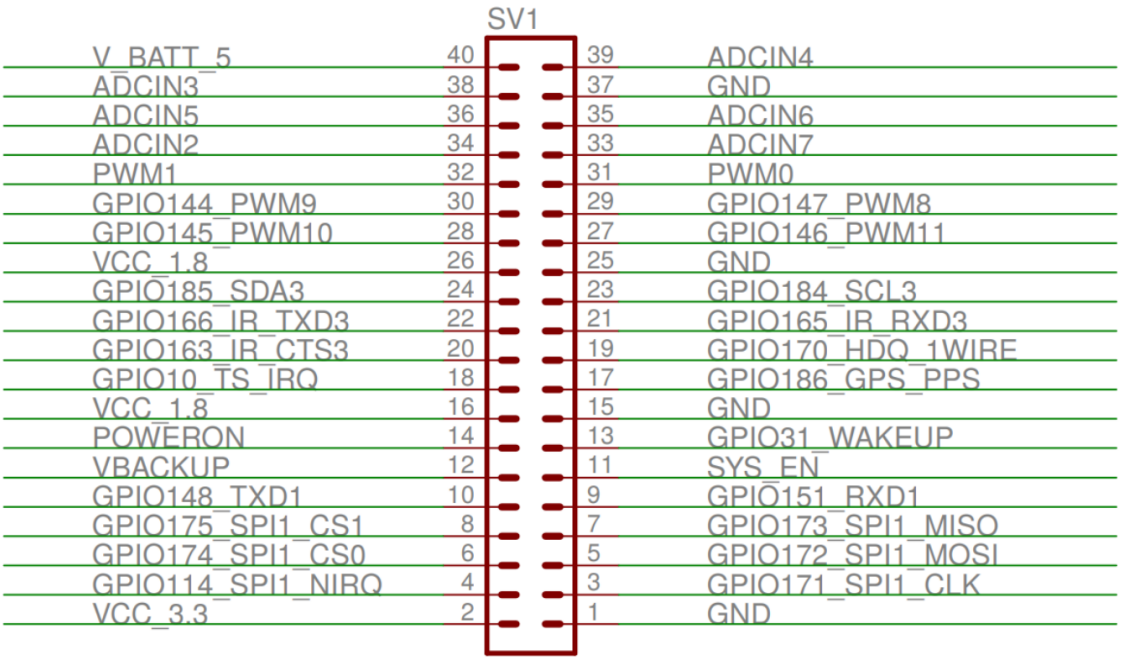


Fig. 6: Diagrama dos pinos da placa de expansão Tobi.

Controle do GPIO via terminal

A maneira mais simples, porém menos eficiente de se controlar o GPIO está descrita no próprio site da fabricante, disponível em Control Overo GPIO . Lá eles indicam controlar o GPIO pelo próprio terminal do sistema Linux através de um sistema *sysfs*. O sistema *sysfs* é um sistema de "arquivos" oferecidos pelo núcleo do Linux para o controle e comunicação com dispositivos e drivers através do terminal do Linux.

Se, por exemplo, desejarmos controlar a saída do **GPIO10** através deste método para piscar um LED precisaremos exportar o **GPIO10** para o espaço do usuário escrevendo `10` no arquivo */sys/class/gpio/export*, o que irá gerar um diretório com outros arquivos para a manipulação do **GPIO10**. Em seguida, devemos definir sua direção como de saída escrevendo `out` em */sys/class/gpio/gpio10/direction* e definir seu valor como alto ou baixo escrevendo `1` ou `0` em */sys/class/gpio/gpio10/value*.

Dica: A função de configuração de interrupção também é acessível pelo terminal.

Este processo pode ser feito tanto pelo terminal do usuário com o comando `echo`, quanto por um programa que abra esse arquivo e escreve nela por nós. Por exemplo, para controlar o **GPIO146** através do terminal podemos executar os seguintes comandos (exemplo utilizado no site da Gumstix):

Nota: Lembrando que o comando `echo teste > pasta/arquivo` irá sobrescrever todo o arquivo pela

palavra «teste» e o comando `cat pasta/arquivo` irá exibir o conteúdo do arquivo.

```
root@overo# echo 146 > /sys/class/gpio/export
root@overo:/sys/class/gpio# cat gpio146/direction
in
root@overo# echo out > /sys/class/gpio/gpio146/direction
root@overo:/sys/class/gpio# cat gpio146/direction
out
root@overo# cat /sys/class/gpio/gpio146/value
0
root@overo# echo 1 > /sys/class/gpio/gpio146/value
root@overo# cat /sys/class/gpio/gpio146/value
1
```

Esse comando controlará o pino 27 da placa Tobi.

Dica: Se você não possuir um medidor, um LED de 1,8V pode ser utilizado. Use o pino 1 como aterramento.

Porém, como já comentado, esse método é bem lento e não pode ser utilizado para comunicação entre dispositivos. Entretanto para atividades com períodos superiores a 100 milissegundos este método pode ser utilizado tranquilamente.

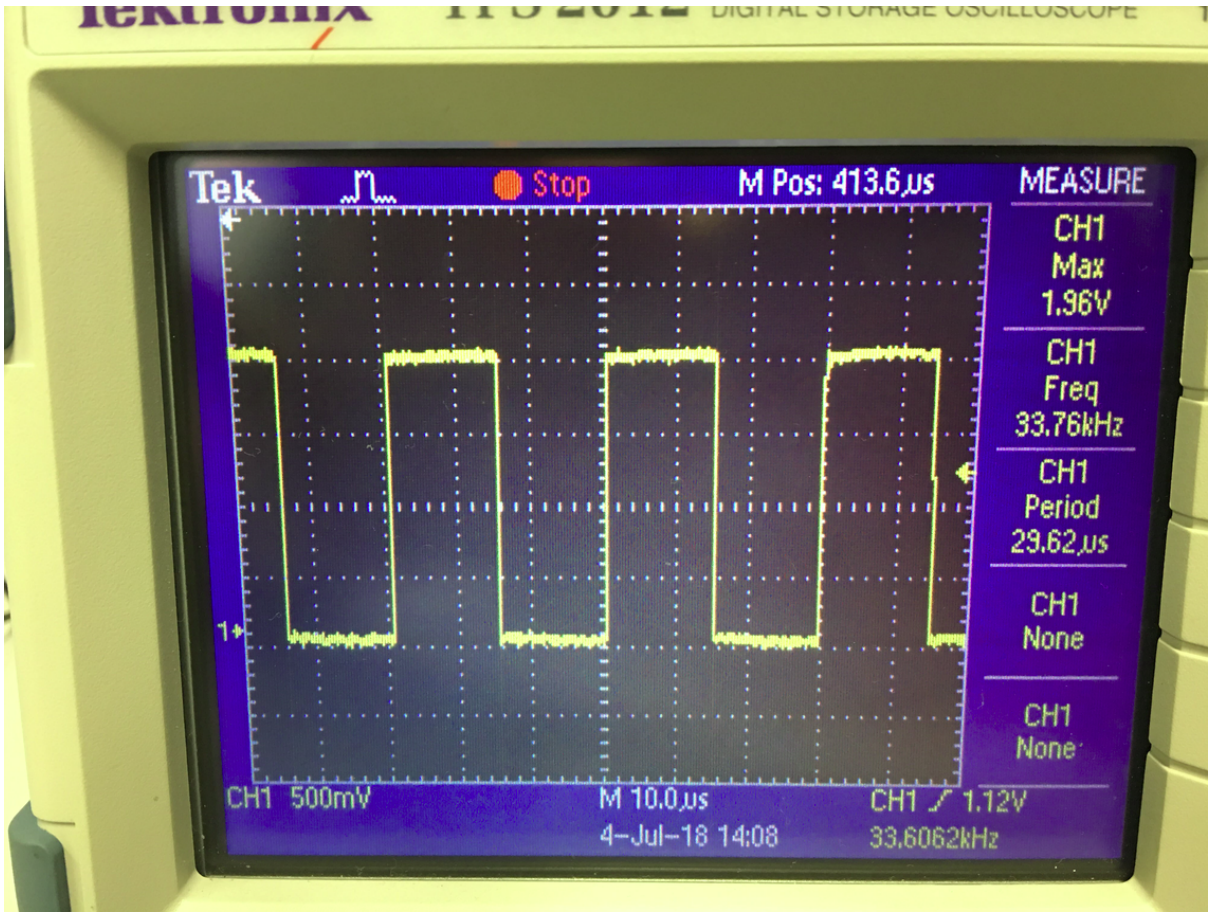
Outra abordagem, utilizando o mesmo método, é utilizar um código semelhante ao código apresentado abaixo, que escreve diretamente nos arquivos do **GPIO**. Essa abordagem foi testada e melhorou consideravelmente, através de um simples código, o tempo de resposta do GPIO.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <termios.h>

int main()
{
    int arq = open("/sys/class/gpio/export", O_WRONLY);
    write(arq, "10", 2);
    close(arq);

    arq = open("/sys/class/gpio/gpio10/direction", O_WRONLY);
    write(arq, "out", 3);
    close(arq);

    arq = open("/sys/class/gpio/gpio10/value", O_RDWR);

    for (int i = 0; i < 10000; i++)
    {
        write(arq, "1", 1);
        //usleep (500000);
        write(arq, "0", 1);
        //usleep (500000) ;
    }
    close(arq);

    return 0;
}
```

Download do código 1 comentado

Para testar o código, o pino 18 (pino do **GPIO** 10) foi conectado a um osciloscópio com o objetivo de medir o período da forma de onda. O resultado dessa medida pode ser visto na figura abaixo, nela podemos ver a

amplitude da forma de onda de 1,96 V, frequência de 33,76 kHz e período de 29,62 microssegundos. Para a maioria das aplicações podemos utilizar esse método.



Controle do GPIO via registradores

Outra maneira de se controlar o GPIO é escrevendo diretamente nos registradores do sistema. Apesar de o procedimento ser um pouco mais complexo essa, na verdade, é a maneira mais comum e mais recomendada de se realizar esse procedimento oferecendo resultados muito mais rápidos.

Para utilizar este método precisamos, primeiro, definir em quais registradores devemos escrever e o que devemos escrever neles. Essa informação só pode ser encontrada no `Technical Reference Manual (TRM)` do processador DM3730, disponivel no site da Texas Instruments.

Como é explicado na seção **25** do TRM do processador DM3730, a partir da página 3477, a interface de controle combina seis bancos de GPIO. Cada modulo de GPIO providencia 32 pinos, totalizando 192 pinos que podem ser utilizados como input e/ou output. Em nosso caso apenas alguns desses 192 pinos estão fisicamente acessíveis, como pode ser visto na figura apresentada abaixo. Cada banco de GPIO possui 26 registradores distribuídos a partir de um endereço de base, sendo que cada um desses registradores possui um comprimento de 32 bits ou 4 bytes.

Nota: A figura foi retirada do *Technical Reference Manual* do processador DM3730 e mostra um pouco mais detalhadamente como esses pinos estão distribuídos entre os módulos dos GPIO. A explicação detalhada de cada um desses registradores pode ser encontrada no manual do processador DM3730.

Neste trabalho apenas dois dos registradores serão comentados de forma a ilustrar o funcionamento desses registradores.

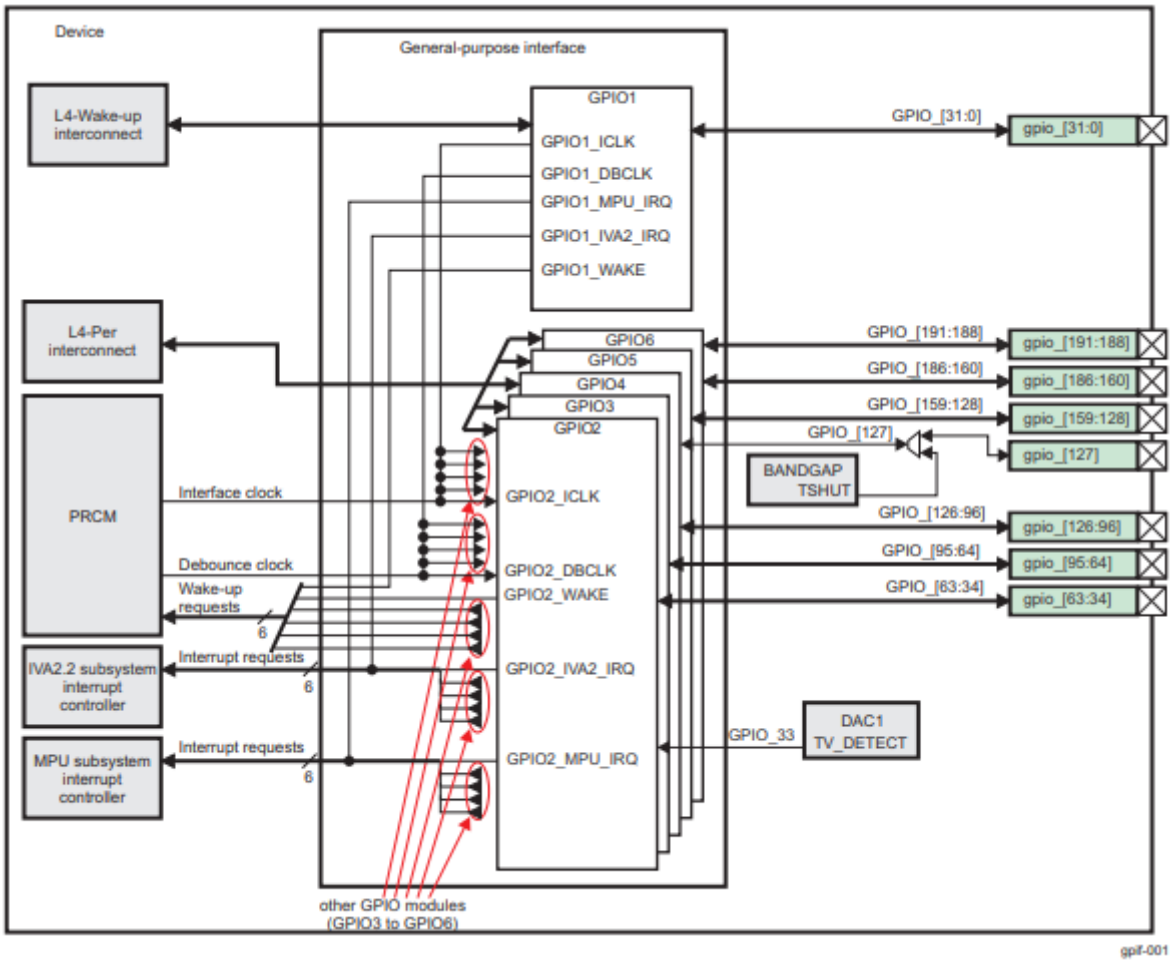


Fig. 7: Diagrama da interface de GPIO.

O registrador **GPIO_OE** é o registrador que define a direção do pino que está sendo configurado. A abreviação «OE» vem de *output enable*. Esse registrador possui um offset de endereço igual a «0x034», ou seja, seu endereço será o endereço de base do modulo do GPIO mais 34 em hexadecimal. Esse registrador possui 32 bits do tipo «Read/White», ou assim, se o pino correspondente à porta GPIO estiver armazenando o valor **0**, essa porta GPIO estará configurada para operar como output, caso neste pino esteja o valor **1** a porta estará configurada como input.

O registrador **GPIO_SETDATAOUT** é o registrador que tem a função de colocar o bit correspondente ao registrador **GPIO_DATAOUT** em 1. Ou seja, se tudo estiver configurado corretamente, surgirá no pino físico o valor de tensão equivalente ao bit 1. Esse registrador possui endereço de offset igual a «0x094». Assim como o registrador comentado anteriormente este registrador é constituído por 32 bits do tipo «RW». A leitura de qualquer um dos bits deste registrador retorna o valor do bit correspondente em **GPIO_DATAOUT**.

Além dos registradores apresentados na seção 25 do *Technical Reference Manual*, também é necessário configurar um registrador do *System Control Module* (**SCM**). O SCM é um módulo que permite o controle através de *software* de várias funções do dispositivo. Para nossa aplicação, o SCM é o ponto primário de controle da função de GPIO e é nele onde vamos realizar a multiplexação, que determina se o pino irá operar na função de GPIO ou em sua função específica, e definiremos se o GPIO será do tipo *pullup* ou *pulldown*, por exemplo.

Os registradores do SCM são divididos em cinco classes. Entretanto, para nossa aplicação iremos utilizar apenas uma, o bloco de registradores de configuração e multiplexação. Esse bloco é um conjunto de registradores de 32 bits, que configura 2 pinos e define, além dos dois parâmetros mencionados anteriormente, a função de *wakeup*. Aos registradores pertencentes a esse bloco é dado o nome de *Configuration Register Functionality*.

Nota: Mais informações sobre o SCM podem ser encontradas na seção 13 do *Technical Reference Manual*.

Para encontrarmos qual o endereço de cada registrador deste tipo podemos procurar na tabela 13-4 do TRM. Nessa tabela será dado o endereço físico exato de cada registrador (base+offset). No caso o endereço base é o próprio endereço dos registradores «PADCONFS» da interface do SCM, encontrado na seção 13.6.1 do TRM e o endereço offset de cada registrador deste bloco pode ser encontrado na tabela 13-73 do mesmo documento.

Após a identificação dos registradores podemos iniciar a elaboração de um código para modifica-los. Assim nos deparamos com mais um desafio, sistemas operacionais trabalham com dois conceitos de memória, memória física e memória virtual. Memória física é a memória do hardware, aquela qual sabemos o endereço e pois verificamos no TRM. Entretanto se criarmos um ponteiro que aponta para a memória «0x4800000», por exemplo, ele não irá apontar para a memória física que possui este endereço pois o sistema operacional mapeia um espaço da memória física diferente para cada programa com os principais objetivos de aumentar a segurança e evitar conflitos de dados entre programas.

Entretanto para ter acesso à memória física do sistema precisamos solicitar ao sistema operacional que mapeie esse espaço de memória para a aplicação. Uma maneira de realizar esse procedimento é através da função «mmap()».

Nota: Detalhes do funcionamento dessa função e seus parâmetro podem ser encontrados em mmap(2) — Linux manual page.

Vamos supor que queremos mapear o espaço de memória físico de «**0x45000000**» até «**0x45001000**» e para isso decidimos usar a função `mmap()`. Portanto, chamamos a função da seguinte maneira, por exemplo, `mmap(NULL,0x1000,PROT_WRITE || PROT_READ,MAP_SHARED,fd,0x45000000)`, executando isso a função irá retornar um ponteiro que aponta para um endereço de memória virtual endereçado no endereço de memória física «**0x45000000**». Em que, para ter acesso à memória física do dispositivo, «**fd**» é o *file descriptor* direcionado para «/dev/mem».

Com essas informações, temos tudo o que é necessário para implementar testes acerca deste modo de operação. A seguir temos um código que aplica o método descrito nesta seção para alternar o nível de tensão do pino «186». Esse código foi implementado para se realizar o mesmo teste da seção «Controle do GPIO via terminal».

Nota: O código abaixo foi obtido no Fórum de Discussões da Gumstix e foram realizadas pequenas alterações para evitar o excesso de informação e facilitar sua compreensão.

```
// Local includes definition
#include <stdio.h>    // for lprint instruction
#include <stdlib.h>
#include <fcntl.h>    // ok for mmap
#include <sys/mman.h> // ok for mmap
#include <unistd.h>

// Defines local parameters (from TRM)
#define SCM_INTERFACE_BASE 0x48002000
#define SCM_PADCONFS_BASE 0x48002030
#define CONTROL_PADCONF_SYS_NIRQ (*(volatile unsigned long *)0x480021E0)
#define CONTROL_PADCONF_SYS_NIRQ_OFFSET 0x1B0

#define GPIO6_BASE 0x49058000
#define GPIO6_SYSCONFIG_OFFSET 0x10
#define GPIO6_CLEARDATAOUT_OFFSET 0x90
#define GPIO6_SETDATAOUT_OFFSET 0x94
#define GPIO6_OE_OFFSET 0x34
#define GPIO6_CTRL_OFFSET 0x30

#define MAP_SIZE (volatile unsigned long)4 * 1024
#define MAP_MASK (volatile unsigned long)(MAP_SIZE - 1)

// Defines "volatile unsigned long" how "u32"
#define u32 volatile unsigned long

// Defines commom variables
u32 *A;
u32 *B;

int main() // Local functions definition
{
    // Defines local variables
    unsigned long i;
    int fd;
    int j;

    fd = open("/dev/mem", O_RDWR | O_SYNC); // "O_RDWR" opens the file for reading␣
↪and writing & "O_SYNC" guarantees that the call will not return before all data␣
↪has been transferred to the disk

    A = (u32 *)mmap(NULL, MAP_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, fd, SCM_
↪INTERFACE_BASE & ~MAP_MASK); // creates a new mapping in the virtual address␣
↪space

    *(u32 *)((u32)A + 0x30 + CONTROL_PADCONF_SYS_NIRQ_OFFSET) |= (0x00040000); //
↪set mode 4 on the pad 186 configuration register; enables digital pin use

    close(fd);
    /********/

    fd = open("/dev/mem", O_RDWR | O_SYNC);

    B = (volatile unsigned long *)mmap(NULL, MAP_SIZE, PROT_READ | PROT_WRITE, MAP_
↪SHARED, fd, GPIO6_BASE & ~MAP_MASK); // COM1 0x4806A000

    //gpio_186 handling
    *(u32 *)((u32)B + GPIO6_SYSCONFIG_OFFSET) |= 0x00000004; // bit2=1 enable/wake␣
↪up, free running clock

    //*(u32 *)((u32)B + GPIO6_CTRL_OFFSET) &= 0xfffffffe; // bit0=0 module enabled,
↪ clock not gated , clock=interface clock divided by 8
```

(continues on next page)

(continuação da página anterior)

```
    *(u32 *)((u32)B+GPIO6_CTRL_OFFSET)&= 0xfffffff8;  // bit0=0,bit1=0,bit2=0␣
↪module enabled, clock not gated , clock=interface clock not divided

    *(u32 *)((u32)B + GPIO6_OE_OFFSET) &= 0xfbffffff; // bit26=0, gpio_186 output

    // generate a pulse stream on gpio_186 pin output

    for (j = 0; j < 1000000; j++)
    {
        *(u32 *)((u32)B + (GPIO6_CLEARDATAOUT_OFFSET)) |= 0x04000000;
        //printf("Saida = 0\n");
        //usleep(1000000);


        *(u32 *)((u32)B + (GPIO6_SETDATAOUT_OFFSET)) |= 0x04000000;
        //printf("Saida = 1\n");
        //usleep(1000000);
    }
    close(fd);
    return (0);
}
```

Download do código 2 comentado

O código acima foi testado da mesma maneira que o código apresentado na seção anterior. Já na figura a seguir é possível ver o resultado deste teste. Observe que dessa vez o tempo obtido foi 720,3 nano segundos, ou seja, aproximadamente 42 vezes mais rápido que o resultado do outro método. Além disso, podemos observar que a forma de onda não é mais um sinal retangular exato, a presença de um efeito capacitivo retardando o processo é evidente, portanto, é possível que essa seja a velocidade máxima em que o sinal de um pino pode ser alterado.

Muito dificilmente alguma aplicação envolvendo GPIO não será satisfeita por algum dos métodos aqui apresentados.

Problemas de escrita em registradores

Para finalizar este último tópico é necessário destacar alguns problemas recentemente encontrados envolvendo escrita em registradores.

O primeiro problema encontrado ocorre sempre que tentamos alterar o valor dos registradores «**0x49050030**», «**0x49056030**» e «**0x49058030**», responsáveis por controlar o clock de todo o bloco do «**GPIO_2**», «**GPIO_5**» e «**GPIO_6**», respectivamente.

Nota: `devmem2` é um comando que executa um programa simples para ler ou escrever em qualquer espaço de memoria. Mais informações podem ser encontradas em devmem2 - Ubuntu Manual.

O que ocorre é que instantes após a alteração do valor do registrador, seu valor retorna ao que possuía antes de ser alterado. Como o teste desta seção apresentou frequência muito alta ele não foi interrompido por este efeito, porém o fenômeno ocorre inclusive quando alteramos valores dos registradores por comandos do terminal, como o `devmem2`. Esse problema está exemplificado na figura abaixo, onde executamos o comando `devmem2 0x49058030 w 0x2` para modificar o registrador **0x49058030** que é o registrador que controla o clock de todo o bloco do **GPIO6**.

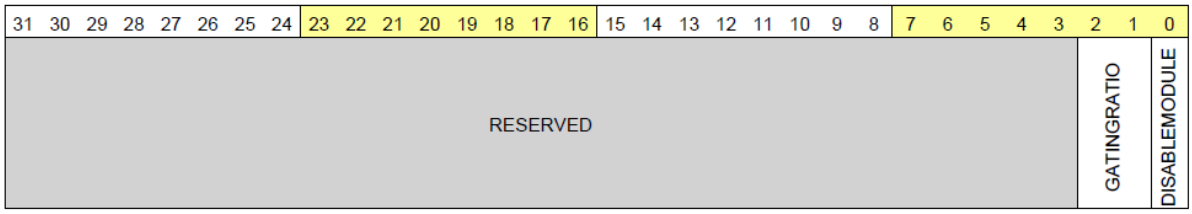
Tal modificação deveria realizar uma redução na velocidade do clock dividindo-o por 2, como indicado no Technical Reference Manual (TRM) do processador DM3730, na tabela 25-29, página 3528, onde é explicado que o **GPIO_CTRL** pode ter seu clock dividido por certos valores pré-cadastrados, como apresentado na figura a seguir.

Porém, logo após a execução do comando é realizado um procedimento de leitura que garante que tudo foi escrito no registrador como o esperado. No entanto, o mesmo comando, executado instantes depois no modo de leitura,

```
root@overo:~# devmem2 0x49050030
/dev/mem opened.
Memory mapped at address 0xb6ff5000.
Read at address  0x49050030 (0xb6ff5030): 0x00000002
root@overo:~# devmem2 0x49050030 w 0x1
/dev/mem opened.
Memory mapped at address 0xb6fef000.
Read at address  0x49050030 (0xb6fef030): 0x00000002
Write at address 0x49050030 (0xb6fef030): 0x00000001, readback 0x00000001
root@overo:~# devmem2 0x49050030
/dev/mem opened.
Memory mapped at address 0xb6fad000.
Read at address  0x49050030 (0xb6fad030): 0x00000002
root@overo:~#
```


Table 25-29. GPIO_CTRL

Address Offset	0x030		
Physical Address	0x4831 0030	**Instance**	GPIO1
	0x4905 0030		GPIO2
	0x4905 2030		GPIO3
	0x4905 4030		GPIO4
	0x4905 6030		GPIO5
	0x4905 8030		GPIO6
Description	This register controls the clock gating functionality.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:3	RESERVED	Read returns 0	RW	0x00000000
2:1	GATINGRATIO	Gating Ratio	RW	0x1
		0x0: Functional clock is interface clock.		
		0x1: Functional clock is interface clock divided by 2.		
		0x2: Functional clock is interface clock divided by 4.		
		0x3: Functional clock is interface clock divided by 8.		
0	DISABLEMODULE	Module Disable	RW	0x0
		0x0: Module is enabled, clocks are not gated		
		0x1: Module is disabled, clocks are gated		

sempre retorna ao valor anteriormente armazenado, o valor existente no registrador antes da modificação. Vale ressaltar que este problema não ocorre para o método de controle do GPIO via terminal, este método opera até que receba uma ordem de parada do usuário.

O segundo problema encontrado ocorre quando tentamos alterar o valor dos registradores **0x49052030** e **0x49054030**, responsáveis por controlar o clock de todo o bloco do **GPIO_3** e do **GPIO_4**, respectivamente. Nesses registradores em específico, ao tentar executar o comando `devmem2` para alterar o clock de um determinado bloco de GPIO ou apenas realizar uma leitura, o sistema retorna o erro «*bus error*» como apresentado na figura abaixo, onde executamos o mesmo comando no registrador **0x49054030**.

```
root@overo:~# devmem2 0x49054030
/dev/mem opened.
Memory mapped at address 0xb6f67[  834.972076] Unhandled fault: external abort on non-linefetch (0x101
8) at 0xb6f67030
[  834.983886] In-band Error seen by MPU  at address 0
[  834.989044] ------------[ cut here ]------------
[  834.993896] WARNING: CPU: 0 PID: 1857 at /home/lucas/yocto/build/tmp/work-shared/overo/kernel-sourc
e/drivers/bus/omap_l3_smx.c:162 omap3_l3_app_irq+0xdc/0x130()
[  835.008972] Modules linked in: snd_soc_omap_twl4030 snd_soc_omap_mcbsp snd_soc_omap snd_pcm_dmaengi
ne snd_soc_twl4030 snd_soc_core snd_compress snd_pcm snd_timer snd soundcore twl4030_madc industrialio
 mt9v032 v4l2_common videodev edt_ft5x06 media usb_f_acm u_serial usb_f_ecm g_cdc u_ether libcomposite
 ipv6
[  835.037780] CPU: 0 PID: 1857 Comm: devmem2 Tainted: G        W      3.18.21-custom #1
[  835.046051] [<c00159b4>] (unwind_backtrace) from [<c001213c>] (show_stack+0x10/0x14)
[  835.054199] [<c001213c>] (show_stack) from [<c05f7b4c>] (dump_stack+0x84/0x9c)
[  835.061828] [<c05f7b4c>] (dump_stack) from [<c00427f4>] (warn_slowpath_common+0x6c/0x90)
[  835.070343] [<c00427f4>] (warn_slowpath_common) from [<c00428b4>] (warn_slowpath_null+0x1c/0x24)
[  835.079559] [<c00428b4>] (warn_slowpath_null) from [<c0335ef4>] (omap3_l3_app_irq+0xdc/0x130)
[  835.088562] [<c0335ef4>] (omap3_l3_app_irq) from [<c008fedc>] (handle_irq_event_percpu+0x3c/0x1d8)
[  835.097991] [<c008fedc>] (handle_irq_event_percpu) from [<c00900b4>] (handle_irq_event+0x3c/0x5c)
[  835.107330] [<c00900b4>] (handle_irq_event) from [<c0092c84>] (handle_level_irq+0xb4/0x144)
[  835.116119] [<c0092c84>] (handle_level_irq) from [<c008f5b4>] (generic_handle_irq+0x28/0x3c)
[  835.124969] [<c008f5b4>] (generic_handle_irq) from [<c008f898>] (__handle_domain_irq+0x64/0xc8)
[  835.134124] [<c008f898>] (__handle_domain_irq) from [<c00086c0>] (omap_intc_handle_irq+0xb4/0xc4)
[  835.143463] [<c00086c0>] (omap_intc_handle_irq) from [<c05fed24>] (__irq_svc+0x44/0x5c)
[  835.151885] Exception stack(0xdd14df50 to 0xdd14df98)
[  835.157196] df40:                                     00000001 00000001 00000000 de31c2c0
[  835.165802] df60: 00000001 dd14c018 00000000 dd14dfb0 dd14c000 dd14c000 b6f67030 bee80db4
[  835.174377] df80: 00000000 dd14df98 c0083434 c0011aec 200d0113 ffffffff
[  835.181335] [<c05fed24>] (__irq_svc) from [<c0011aec>] (do_work_pending+0x5c/0xc4)
[  835.189300] [<c0011aec>] (do_work_pending) from [<c000e6c0>] (work_pending+0xc/0x20)
[  835.197448] ---[ end trace d18b1a8ec835e19d ]---
000.
Bus error
root@overo:~#
```

Dessa forma, foi possível apenas alterar o clock do bloco do **GPIO_1**, como pode ser visto na imagem abaixo.

```
root@overo:~# devmem2 0x48310030
/dev/mem opened.
Memory mapped at address 0xb6fa6000.
Read at address  0x48310030 (0xb6fa6030): 0x00000000
root@overo:~# devmem2 0x48310030 w 0x1
/dev/mem opened.
Memory mapped at address 0xb6fa9000.
Read at address  0x48310030 (0xb6fa9030): 0x00000000
Write at address 0x48310030 (0xb6fa9030): 0x00000001, readback 0x00000001
root@overo:~# devmem2 0x48310030
/dev/mem opened.
Memory mapped at address 0xb6faa000.
Read at address  0x48310030 (0xb6faa030): 0x00000001
root@overo:~#
```

Não sabemos por quais motivos esses fenômenos estão ocorrendo com os blocos de 2 a 6, porém suspeitasse que alguns processos do sistema operacional estejam impedindo que o clock de tais blocos seja alterados, provavelmente por algum circuito interno ou operação depende de tais valores pré-definidos ou até por alguma restrição no consumo de energia.

Referências

- PITA, H. C. Desenvolvimento de sistema de comunicação multiplataforma para veículos aéreos de asa fixa. Faculdade de Tecnologia, Universidade de Brasília, 2018.
- TEXAS INSTRUMENTS. AM/DM37x Multimedia Device Technical Reference Manual. 12500 TI Blvd, Dallas, TX 75243, EUA, 2012. Version R. Disponível em: ti.com.
- Direct register access control of GPIO ARM interface on Overo Water +TOBI - Gumstix Discussion Forum

Comunicação Serial

A principal característica da comunicação serial é o processo de enviar dados um bit de cada vez, de forma sequencial, através de um canal de comunicação ou barramento. Diferente da comunicação paralela, em que todos os bits de cada símbolo são enviados juntos.

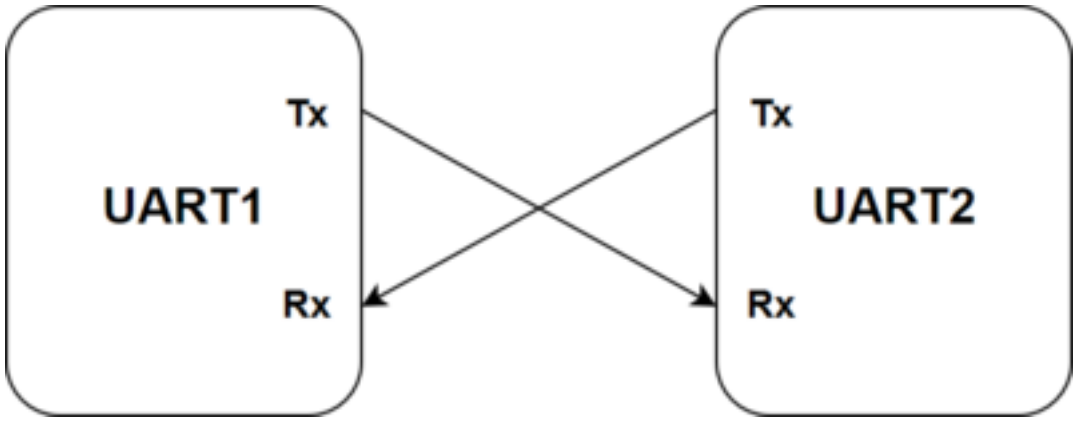
Com a finalidade de possibilitar a conversão, transmissão e recepção de dados de forma serial, sendo estes originalmente dispostos de maneira paralela, surge o formato **UART**, acrônimo de *Universal Asynchrounous Receiver/Transmiter* ou Receptor/Transmissor Universal Assíncrono.

O termo «Universal» refere-se a característica do formato dos dados e velocidade serem configuráveis, enquanto «Assíncrono» diz respeito a forma de comunicação em série ocorre, em que os dispositivos não estão continuamente sincronizadas por um sinal de clock comum.

Nota: Mais detalhes sobre a comunicação UART podem ser vistos em Introduction to UART Communication - microcontrollerslab.com.

No caso do Overo, temos três sistemas de Comunicação **UART** implementados por hardware à nossa disposição. O que faz desnecessário qualquer implementação manual através de software utilizando GPIO.

Agora vamos entender como funciona o protocolo de comunicação UART. Essa comunicação funciona através da conexão do transmissor (**TX**) de um dispositivo com o receptor (**RX**) de outro dispositivo, no caso, apenas o **TX** realiza alterações no nível de tensão da linha, sendo assim, a comunicação é, para cada conexão, uma via de mão única. Logo, para realizar uma comunicação de mão dupla iremos utilizar duas conexões, uma será a ligação de **RX** do dispositivo 1 com o **TX** do dispositivo 2 e a outra ligação será o oposto, **RX** do dispositivo 2 com **TX** do dispositivo 1, similar a imagem abaixo.



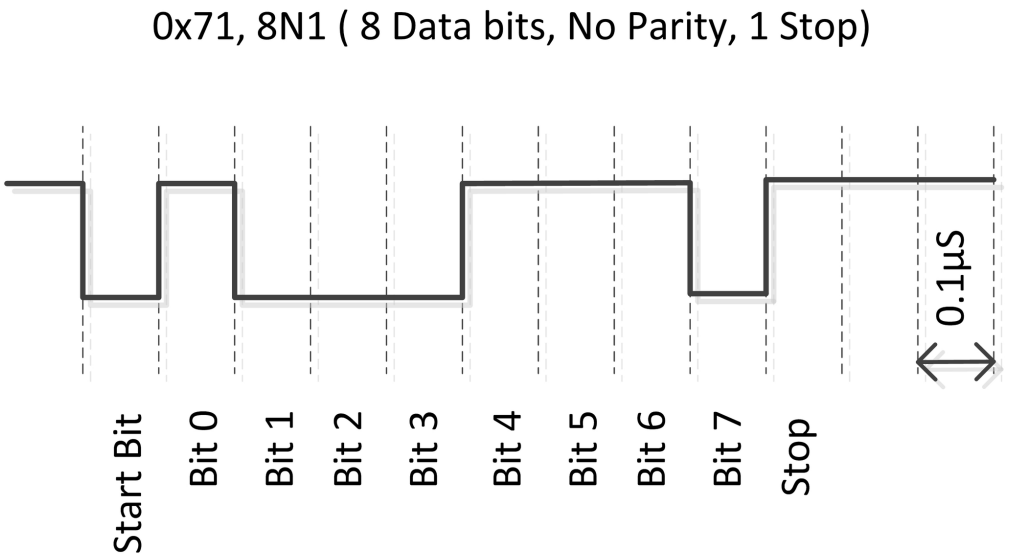
Podemos analisar a situação da comunicação a nível de bit. Por exemplo, para uma comunicação UART do tipo «8N1» (8 bits de dados, 0 bits de paridade e 1 bit de parada) teremos o canal em estado *IDLE*, que significa «não operando», representado pelo nível de tensão estático em alto. Logo, quando pretende-se iniciar a comunicação é enviado um pulso de nível baixo e em seguida são enviados os oito bits de dados que será acompanhado de um bit de parada em estado alto.

É importante destacar que a função do bit de parada é realizar uma pausa na transmissão para algum processamento interno dos dispositivos, não é necessário, portanto, nenhum tempo adicional entre os dados transmitidos.

Como essa é uma comunicação assíncrona, é essencial que a velocidade da comunicação seja pré-determinada. Essa velocidade geralmente é dada em Baud rate, uma unidade de medida para a quantidade de unidades de sinal enviadas por segundo.

Nota: O termo «Baud rate» é utilizado como medida de velocidade de transmissão de dados entre dispositivos. Um baud é uma medida de velocidade de sinalização e representa o número de mudanças na linha de transmissão (seja em frequência, amplitude, fase etc. . .) ou eventos por segundo.

A figura a seguir apresenta um esquemático que ilustra o funcionamento da comunicação UART. Na ilustração a velocidade de comunicação é de 10.000.000 baud/s.



Particularidades da Gumstix Overo

Por padrão, a placa de expansão Tobi disponibiliza apenas dois dos UARTs disponíveis no computador em modulo Overo para uso através dos seus pinos. Como pode ser visto na imagem abaixo, a UART1 está conectada aos pinos 10 e 9 e a UART3 está conectada aos pinos 22 e 21. Ainda é importante dizer que a porta serial UART3 é o mesmo pino utilizado pelo «USB console», ou seja, é o mesmo pino que usamos para controlar a Gumstix pelo computador, ou seja, em configuração padrão as mensagens do sistema e as mensagens para o sistema são enviadas por esta porta.

Caso as duas portas seriais já mencionadas não sejam suficientes, existe ainda a porta serial UART2. Porém, por padrão, está não está disponível em nenhum dos pinos da placa Tobi para nossa utilização. Na verdade, ela foi reservada para comunicar-se com o Bluetooth, contudo apenas versões posteriores do computador embarcado Overo por nos utilizado possuem Bluetooth, portanto podemos, caso necessário, exportar a UART2 para os pinos e utiliza-los. Para utilizar esta porta serial é necessário modificar o *u-boot* de modo a multiplexar sua função ao GPIO 146/147 que, como mostrado na figura anterior, estão ligados aos pinos 29 e 27. Portanto para fazê-lo é necessário modificar o arquivo «overo.h» removendo as linhas de comando referentes ao modo de GPIO dos pinos 146 e 147, remover as linhas que desabilitam a UART2 e adicionar as linhas que habilitam a comunicação serial pela UART2.

Para entender, detalhadamente, o que precisa ser feito e quais registradores serão alterados deve-se consultar a seção de comunicação serial do Technical Reference Manual (TRM) do processador. Contudo existe um tópico no Fórum de Discussões da Gumstix que indica diretamente quais alterações devem ser feitos no «u-boot» para que se possa utilizar a UART2, apesar disso a solução apresentada nesse fórum não foi testada durante este trabalho.

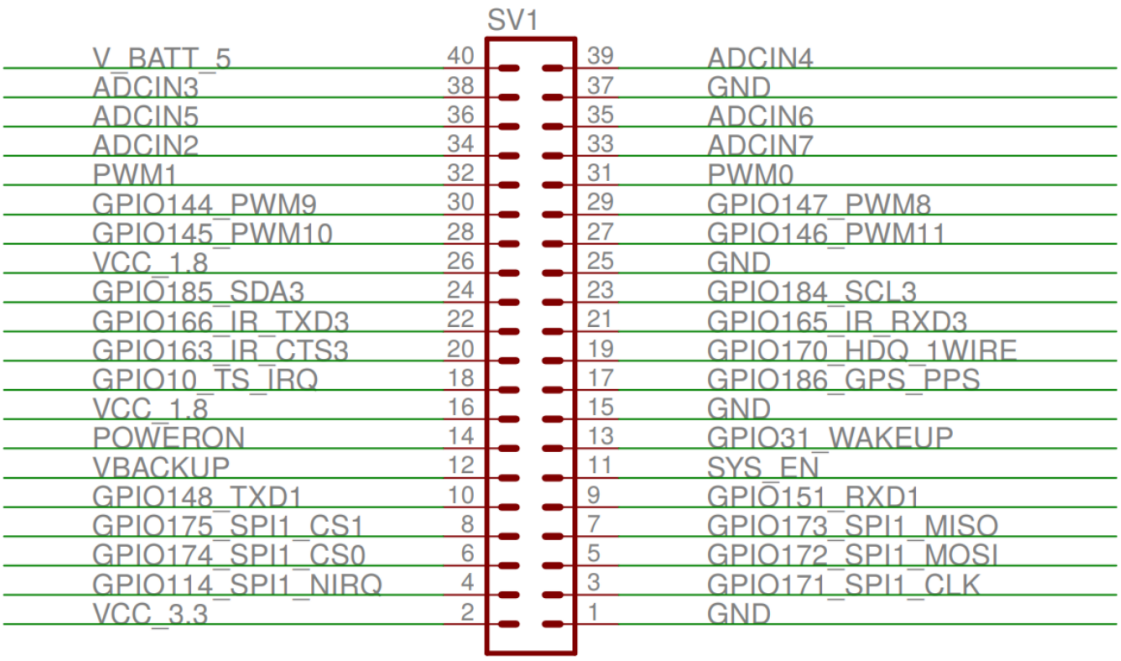


Fig. 8: Diagrama dos pinos da placa de expansão tobi.

Configuração da UART

Como já comentado anteriormente, o computador embarcado possui um hardware específico para comunicação UART, ou seja, não é necessário realizar uma implementação manual, para utilizar a comunicação UART basta escrever em alguns registradores para enviar a mensagem.

Na verdade, em nosso caso é ainda mais simples pois o sistema operacional instalado já traz configurado drivers para a aplicação da comunicação serial. Portanto, não é necessário acessar a memória física do dispositivo, precisamos apenas escrever no driver o que deve ser transmitido.

Os drivers de comunicação serial são arquivos do tipo caractere com nome «**ttyOx**», em que «**x**» é um número exclusivo para cada uma das UARTs. Esses drivers estão localizados em «**/dev**» e funcionam como comunicação em terminal.

Por exemplo, o driver «**ttyO2**» é o driver de comunicação serial da porta «**USB Console**» a mesma que conectamos ao computador, ou seja, ao escrever nessa porta escreveremos no computador conectado à Gumstix e ao ler essa porta estaremos lendo o computador. Em outras palavras, escrever ou ler nesse driver terá o mesmo resultado final de chamar, respectivamente, a função `printf()` ou `scanf()`, quando um computador estiver conectado a essa porta com o terminal aberto.

A configuração das portas seriais pode ser feita de duas maneiras, por linhas de comando no terminal Linux ou por um código que altere as configurações do hardware. A mais simples e, novamente, mais limitada ou menos eficiente é a configuração por meio de linhas de comando, a configuração por esse modo costuma ser usada apenas quando feita por um usuário humano em tempo real.

Para realizar a configuração por meio do terminal Linux devemos utilizar o comando `stty`, já que esse comando possui uma enorme quantidade de parâmetros que permite estabelecer a comunicação serial da forma desejada.

Nota: Para visualizar todos os parâmetros do comando `stty` basta executar `stty -help` no terminal.

Se, por exemplo, for executada a linha de comando `stty -F /dev/ttyO0 -a` serão impressas todas as configurações da comunicação serial **UART1** do dispositivo. Para imprimir apenas as principais configurações, deve-se suprimir o `-a`, a última opção do comando. Caso a alteração da velocidade seja desejável, ela pode ser alterada simplesmente acrescentando a velocidade desejada ao final da linha de comando.

A figura abaixo apresenta um exemplo de configuração da UART1 por meio do terminal de comandos Linux.

```
root@overo:~# stty -F /dev/ttyO0
speed 9600 baud; line = 0;
-brkint -imaxbel
root@overo:~# stty -F /dev/ttyO0 raw 115200
root@overo:~# stty -F /dev/ttyO0
speed 115200 baud; line = 0;
min = 1; time = 0;
-brkint -icrnl -imaxbel
-opost
-isig -icanon
root@overo:~#
```

A outra maneira de configurar a comunicação serial feita por esses drivers sem alterar manualmente o conteúdo do endereço físico da memória é com o auxílio da biblioteca «**termios.h**». Essa biblioteca possui uma ampla variedade de funções que configuram a comunicação serial com base nos parâmetros de uma estrutura «**termios**», também definida nesta biblioteca.

Nota: Mais informações sobre a biblioteca «**termios.h**» podem ser encontrados em termios.h - Linux manual page.

São dois os parâmetros da comunicação UART, além dos mencionados anteriormente, que se destacam, o número mínimo de bits que se espera ler em cada tentativa de leitura e o tempo máximo de espera por um novo caractere após a transmissão do último caractere e após o número mínimo de caracteres ser atingido.

O número mínimo de bits que se espera ser lido e o tempo máximo de espera pelo próximo bit em décimos de segundo podem ser configurados com os seguintes comandos `termios.c_cc[VMIN] =` e `termios.c_cc[VTIME] =`, em que `termios` é o nome de sua estrutura. Para a configuração de velocidade recomenda-se usar a função `cfsetspeed()`.Já a função `cfmakeraw()` configura, além de outros parâmetros, o funcionamento sem bit de paridade e com 8 bits de dados. Após realizados os ajustes na estrutura é necessário ainda executar a função `cfsetattr()` para que as alterações sejam feitas na UART.

Abaixo encontra-se o código utilizado para configurar a comunicação serial dos computadores Overo. Observe que nessa função de configuração não foi utilizada a flag «O_NONBLOCK» na função «open()» e foi definido como 1 o número mínimo de caracteres a serem retornados após uma tentativa de leitura, portanto caso o código seja executado e nenhuma informação seja enviada para este canal o processador aguardará eternamente por esse caractere. A contagem de tempo, definida como 0,1 segundo, só inicia após o número mínimo de caracteres ser atingido.

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <termios.h>

void main()
{
    struct termios cUART1;
    int UART1 = open("/dev/ttyO0", O_RDWR);

    if(tcgetattr(UART1,&cUART1))
        printf("Erro tcgetattr");
    cfmakeraw(&cUART1);
    cfsetspeed(&cUART1,B115200);
    cUART1.c_cflag &= ~CSTOPB;

    cUART1.c_cc[VMIN] = 1;
    cUART1.c_cc[VTIME] = 1;
```

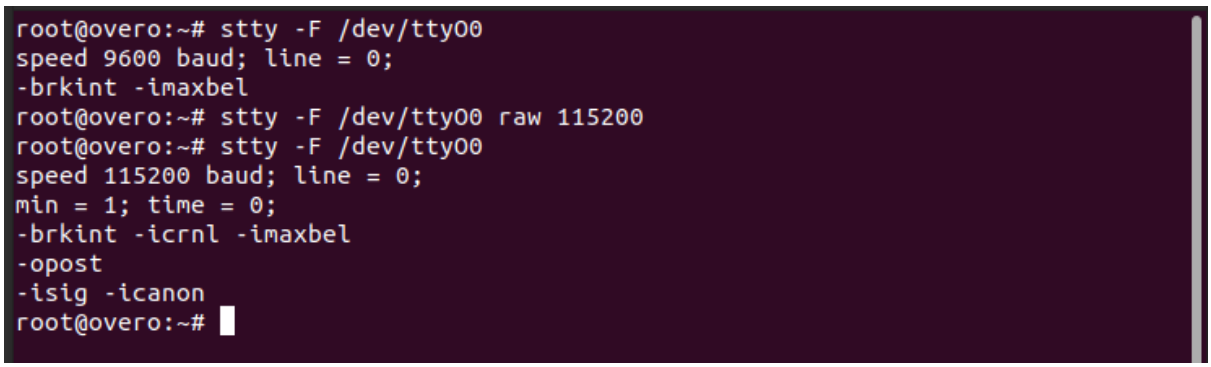
(continues on next page)

(continuação da página anterior)

```
    if (tcsetattr(UART1, TCSANOW, &cUART1))
        printf("Erro tcsetattr");

}
```

Download do código comentado

A figura a seguir apresenta um exemplo de configuração da UART1 por meio do codigo de configuração acima.



Nota: Com a finalidade simplificar a configuração do UART dentro de um outro código, foram efetuadas algumas modificações no código anterior para converte-lo em uma função para configuração de comunicação serial, como pode ser visto abaixo:

```
int configUART1()
{
    struct termios cUART1;
    int UART1 = open("/dev/ttyO0", O_RDWR);

    if(tcgetattr(UART1,&cUART1))
        printf("Erro tcgetattr");
    cfmakeraw(&cUART1);
    cfsetspeed(&cUART1,B115200);
    cUART1.c_cflag &= ~CSTOPB;

    cUART1.c_cc[VMIN] = 1;
    cUART1.c_cc[VTIME] = 1;
    if (tcsetattr(UART1, TCSANOW, &cUART1))
        printf("Erro tcsetattr");

    return UART1;
}
```

Uma vez feita a configuração, foi implementado também o código a seguir com a finalidade de testar a comunicação entre dois computadores. No teste, um dispositivo envia uma mensagem para o outro dispositivo que responde com uma mensagem semelhante para o primeiro dispositivo, em seguida ambos os dispositivos imprimem a mensagem recebida.

```
int main()
{

    int UART1 = configUART1();  // call the UART configuration function
    char dis[2], out[100], string[100];

    printf("Que dispositivo eu sou?");
    scanf("%c", &dis[0]);
    dis[1] = 0;
```

(continues on next page)

(continuação da página anterior)

```
    string[0] = 0;
    strcat(string, "Ola! Essa e uma mensagem do dispositivo ");
    strcat(string, dis);

    // testa UART
    write(UART1, string, strlen(string));
    sleep(1);
    read(UART1, out, 100);
    printf("Mensagem lida pelo dispositivo %s: %s\n", dis, out);
    close(UART1);
    return 0;
}
```

Download do código completo

Como os dois dispositivos são idênticos, será necessário conectar o pino 10 de um dispositivo com o pino 9 do outro dispositivo e vice-versa. Utilizando esse código como base é possível enviar qualquer mensagem de até 100 caracteres de um dispositivo ao outro.

A figura a seguir apresenta o resultado do teste dos códigos apresentados. Nessa figura podemos ver dois terminais do Linux, cada um vinculado a um computador embarcado, e ambos chamam a mesma função, logo em seguida vemos a mensagem lida por cada um dos dispositivos.

Referências

- PITA, H. C. Desenvolvimento de sistema de comunicação multiplataforma para veículos aéreos de asa fixa. Faculdade de Tecnologia, Universidade de Brasília, 2018.
- Universal asynchronous receiver-transmitter - wikipedia.org
- Asynchronous serial communication - wikipedia.org
- Como funcionam as UARTs - newtoncbraga.com.br
- UART Basics - ece353.engr.wisc.edu
- termios.h(0p) - Linux manual page - man7.org
- cfsetspeed(3) - Linux man page - linux.die.net

1.1.5 Referências

- ROCHA, E. M. C. Desenvolvimento de um sistema com veículos aéreos não-tripulados autônomos. Faculdade de Tecnologia, Universidade de Brasília, 2017.
- CORDEIRO, T. F. K. Desenvolvimento de um sistema com veículos aéreos não-tripulados autônomos. Faculdade de Tecnologia, Universidade de Brasília, 2018.
- PITA, H. C. Desenvolvimento de sistema de comunicação multiplataforma para veículos aéreos de asa fixa. Faculdade de Tecnologia, Universidade de Brasília, 2018.
- PX4 Autopilot User Guide - docs.px4.io
- QGroundControl User Guide - qgroundcontrol.com
- Ardupilot Docs - ardupilot.org
- RT-MaG Project - gipsa-lab.fr
- Yocto Project - yoctoproject.org
- Getting Started - Gumstix COM - gumstix.com
- Gumstix, Inc - GitHub - github.com

1.2 VANT com painéis solares

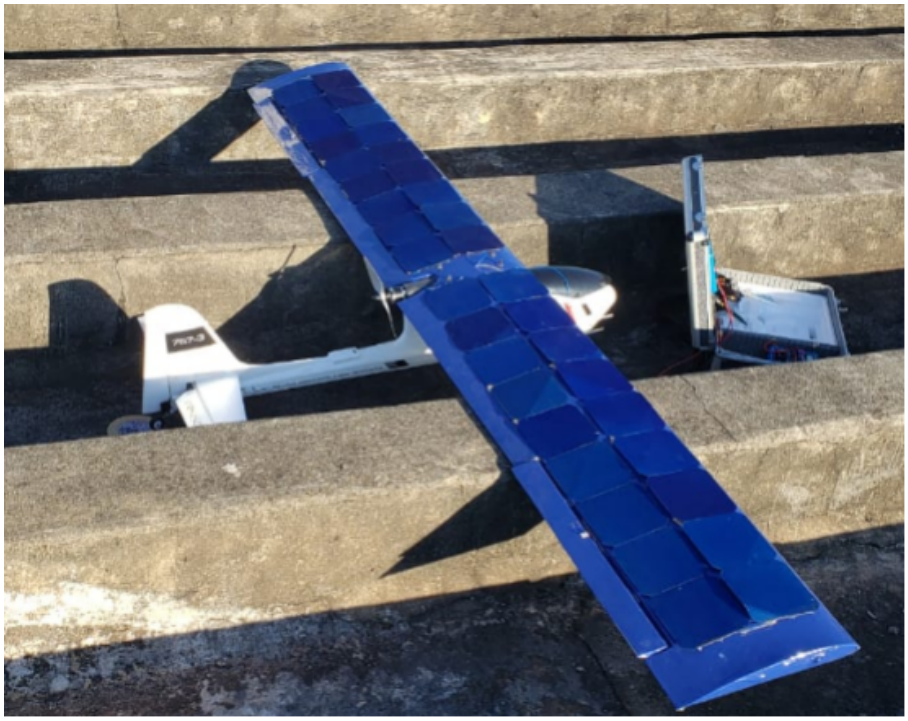


Fig. 9: Aeronave Ranger EX Volatex RC com asa modificada

Este trabalho trata do desenvolvimento de um VANT (veículo aéreo não tripulado) de asa fixa com um sistema de obtenção e gerenciamento de energia por meio de células solares dispostas em sua asa. Os sistemas que compõem o VANT com painéis solares são: Sistema de navegação e controle, sistema de gerenciamento de energia e FPV.

O sistema de gerenciamento de energia permite ao VANT maior autonomia, potencializando sua capacidade em missões de reconhecimento e vigilância. É uilizado neste projeto o algoritmo de rastemaento de ponto de máxima potência que possibilita o ajuste continuo da impedância percebida para que o sistema fotovoltaico opere na região do ponto de máxima potência do painel fotovoltaico.

O sistema de Navegação e Controle possui sensores responsáveis pela determinação da posição, velocidade, altitude, bem como um gerenciamento por software em uma estação em solo para controle e monitoramento da aeronave. Os sensores que compõem o sistema de navegação são: giroscópio, barômetro, magnetômetro, acelerômetro e o GPS.

O VANT com painéis solares é uma modificação do modelo Ranger EX Volatex RC, substituindo-se sua asa original por uma asa com perfil Clark-Y 11,5% smoothed com 36 células fotovoltaicas alocadas na sua superfície.

1.2.1 Visão macro do sistema

Nota: Este topico constém algumas informações disponíveis mais detalhadamente na pagina VR-01.

Introdução

O VANT com painéis solares consiste em um sistema de robótica aerea composto por uma aeronvae radiocontrolada com auxilio de um sistema controle e navegação em conjunto com um sistema de gerenciamento de energia.

O sistema de navegação e controle por sua vez é subdivido em um sistema de malha fechada que realiza a estabilização de vôo e um sistema de telemetria que permite o controle de navegação da aeronave. Por fim, para o

controle manual é utilizado um sistema de FPV (First person view) que consiste em sistema de transmissão de vídeo em tempo real das imagens de uma camera embarcada na aeronave para uma estação em solo.

O sistema de gerenciamento de energia consiste em um conjunto de células fotovoltaicas disposta sobre a asa da aeronvave formando em conjunto um painel solar. Essa energia obtida e gerenciada por um controlador de carga que regula cargas e descargas excessivas do sistema.

Fig. 10: Visão Macro do sistemas do VANT com painéis solares

Conceitos Básicos

Este tópico apresenta alguns conceitos básicos a respeito do VANT com painéis solares.

Veículo aéreo não tripulado (VANT ou *drone*)

Um VANT (Veículo Aéreo Não Tripulado) é um tipo de aeronave em que não há tripulação e é controlada remotamente, podendo realizar vôos em modo autônomos.

Estação de Controle em Solo (ECS)

Uma Estação de Controle em Solo (ECS) é um subsistema de controle na qual uma aplicação de software em um computador em solo se comunica com o VANT fornecendo o comando de execução da missão.

Sistema de gerenciamento de energia (MPPT)

O algoritmo MPPT ((Maximum power point tracking) possibilita o rastreamento do ponto de potência máxima fazendo o ajuste contínuo da impedância percebida para que o sistema fotovoltaico opere no ponto de máxima potência mesmo sob variação da irradiação solar.

FPV

FPV (First person view) é um sistema de transmissão de vídeo de uma camera embarcada na aeronave com observação em primeira pessoa da estação de controle. Sua transmisão é realizada através de receptores na faixa de 5,8 GHz.

Célula Solar e Painel Solar

Célula solar é o nome dado para cada unidade geradora de energia de um painel solar. Neste projeto são placas feitas de silício cristalino que convertem luz solar em energia elétrica. O painel solar por sua vez é associação em série de células solares.

ECT

Célula solar é o nome dado para cada unidade geradora de energia de um painel solar. Neste projeto são placas feitas de silício cristalino que convertem luz solar em energia elétrica. O painel solar por sua vez é associação em série de células solares.

1.2.2 Hardware embarcado

Uma breve descrição dos componentes do VANT Solar é realizada a seguir:

- Célula Solar: A célula solar é um aparato que converte energia proveniente da luz solar em energia elétrica;
- Painel Solar: Associação de células solares a fim de se obter uma quantidade maior de energia obtida atraves da luz solar;
- Controle de carga: São reguladores de cargas e descargas excessivas, mantendo a estabilidade elétrica do sistema. Os dois tipos mais comuns são o sistema PWM (pulse width modulation) e MPPT (maximum power point tracking).
- 3DR Power: Sistema de monitoramento de energia;
- Bateria: Unidade de amarzenamento de energia;
- ESC: Sigla para «Electronic Speed Control», um dispositivo que controla a velocidade de motores CC Brushless;
- PIXHAWK: Unidade de sistema de controle de estabilização de VANT's;
- Servo Motor: Atuador rotativo ou linear que garante o controle de posição em malha fechada;
- Motor Brushless: Motores compostos de imãs permanentes controlado por dispositivo eletrônico de comutação de bobinas elétricas;

A descrição mais detalhada de cada componente pode ser encontrada a seguir:

Célula solar

A célula solar é um aparato que converte energia proveniente da luz solar em energia elétrica. Essas células são denominadas células fotovoltaícas. No projeto do VANT com painéis solares, ele é utilizada como artifício para melhora da autonomia de vôo da aeronave.

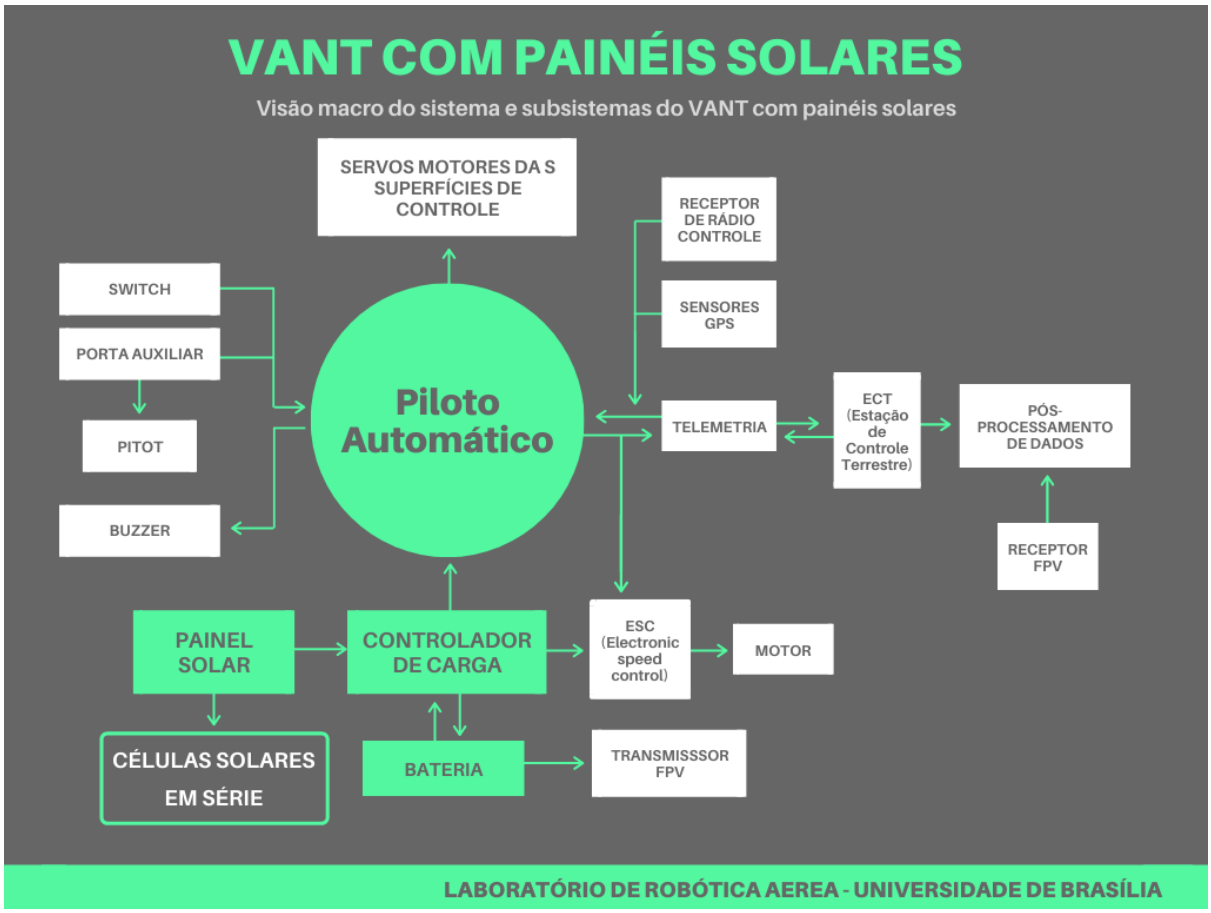


Fig. 11: Diagrama do VANT com painéis solares

Fig. 12: Célula Solar C60 - Fonte: Datasheet SUNPOWER C60 SOLAR CELL

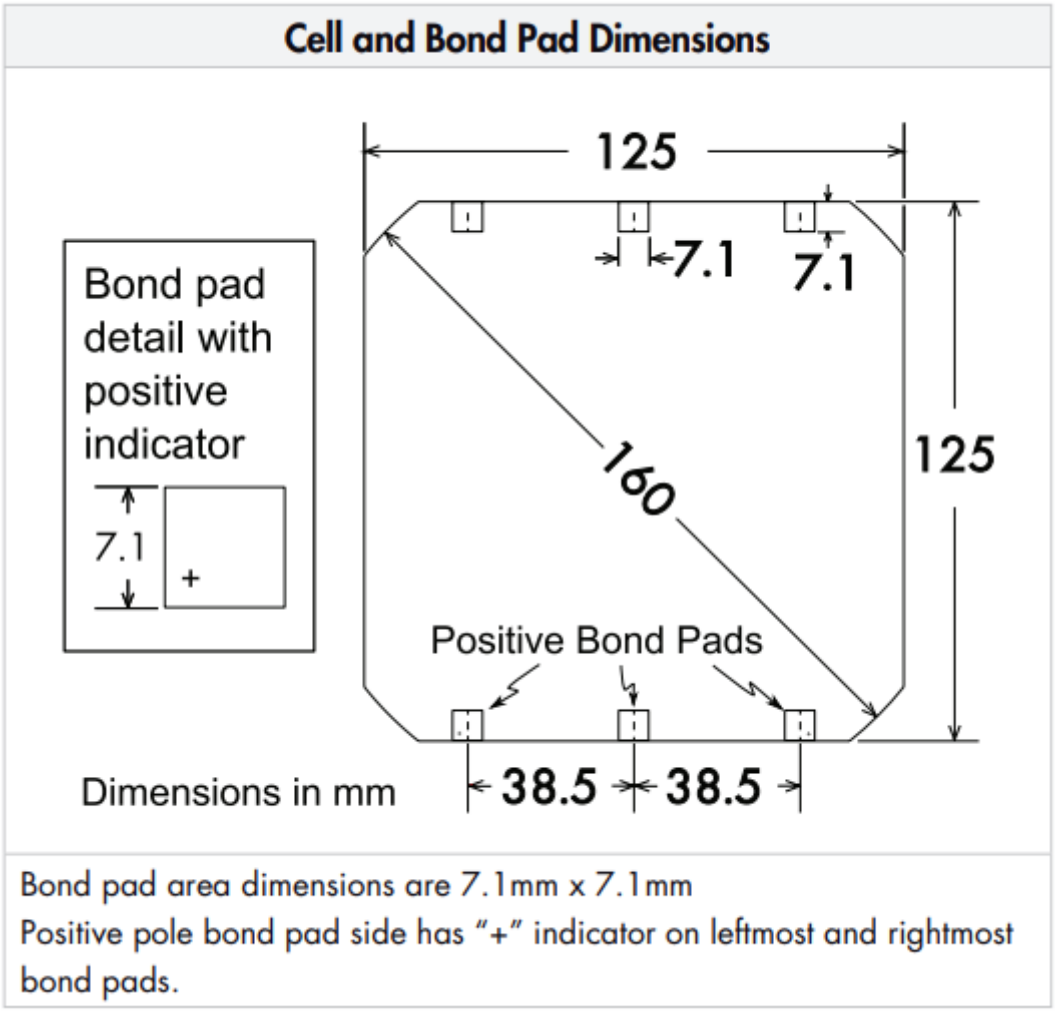


Fig. 13: Dimensões da célula solar C60 - Fonte: Datasheet SUNPOWER C60 SOLAR CELL

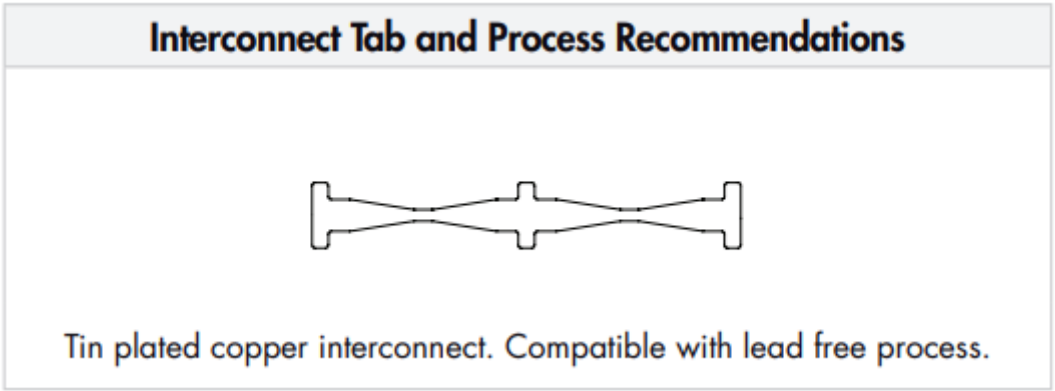


Fig. 14: Conexão da célula solar - Fonte: Datasheet SUNPOWER C60 SOLAR CELL

Electrical Characteristics of Typical Cell at Standard Test Conditions (STC)

STC: 1000W/m², AM 1.5g and cell temp 25°C

Bin	Pmpp (Wp)	Eff. (%)	Vmpp (V)	Impp (A)	Voc (V)	Isc (A)
G	3.34	21.8	0.574	5.83	0.682	6.24
H	3.38	22.1	0.577	5.87	0.684	6.26
I	3.40	22.3	0.581	5.90	0.686	6.27
J	3.42	22.5	0.582	5.93	0.687	6.28

All Electrical Characteristics parameters are nominal
Unlaminated Cell Temperature Coefficients
Voltage: -1.8 mV / °C Power: -0.32% / °C

Fig. 15: Características elétricas da célula solar C60 - Fonte: Datasheet SUNPOWER C60 SOLAR CELL

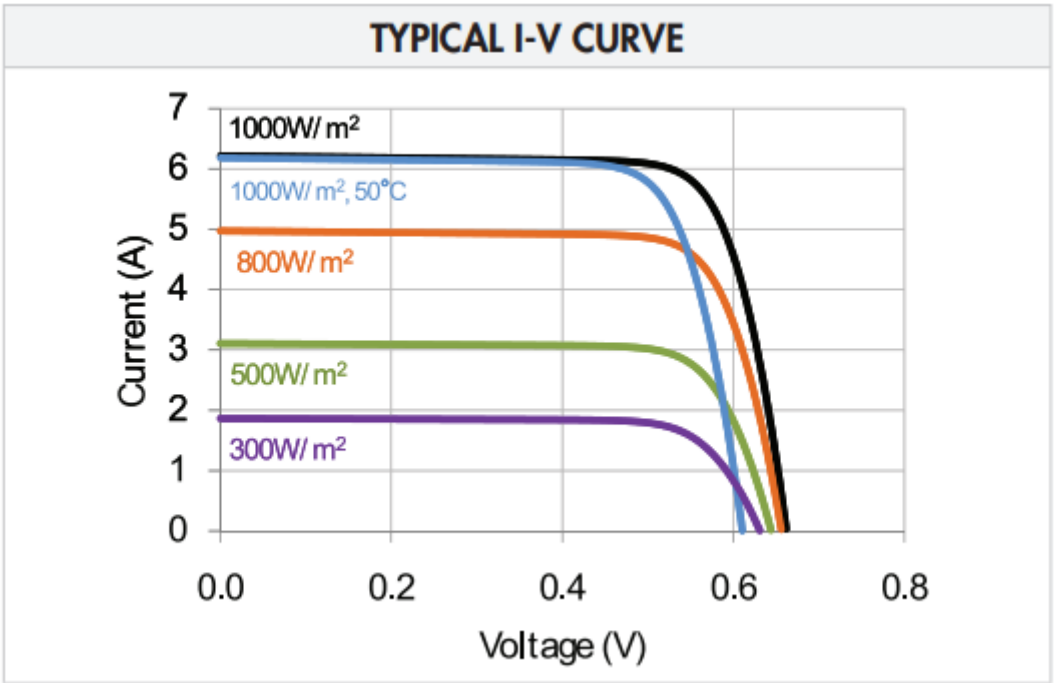


Fig. 16: Típica I-V curva - Fonte: Datasheet SUNPOWER C60 SOLAR CELL

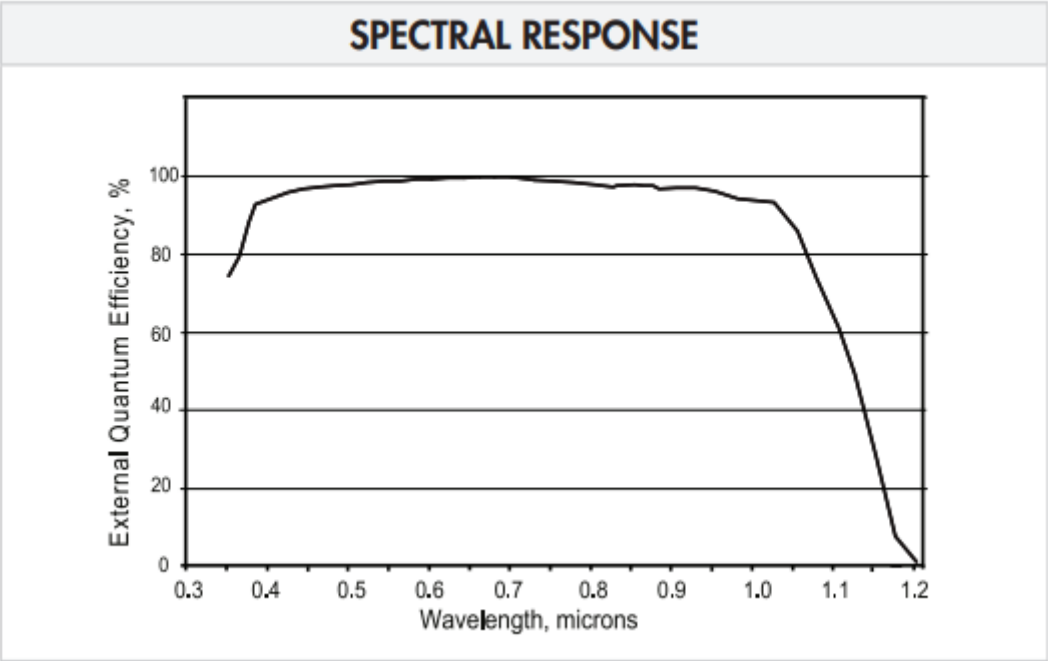


Fig. 17: Curva de eficiência do sensor - Fonte: Datasheet SUNPOWER C60 SOLAR CELL

As células solares utilizadas possuem as seguintes características:

- Vmpp: Tensão de máxima potência;
- Voc: Tensão de circuito aberto;
- Impp: Conrrente de máxima potência;
- Isc: Corrente de de curto circuito;

Table 2: Especificações da célula Maxeon C60

Características	Valor
Dimensões	125 x 125 mm
Eficiência	21,8%
Vmpp	0,574V
Voc	0,682V
Impp	5,83A
Isc	6,24A

Nota: As especificações completas da célula solar estão disponíveis nas fichas técnicas abaixo.

- `Datasheet - C60 Célula Solar`

Controlador de carga

Nota: As especificações completas da célula solar estão disponíveis nas fichas técnicas abaixo.

- `Genasun GV-10 Datasheet`
- `Genasun GV-10 Manual`

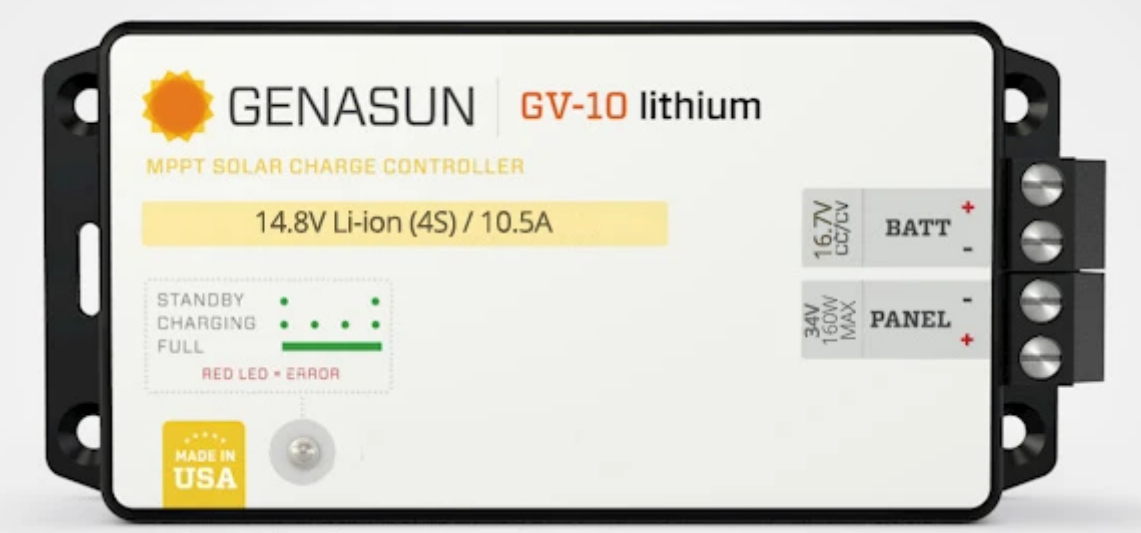


Fig. 18: Controlador de Carga - Fonte: Genasun

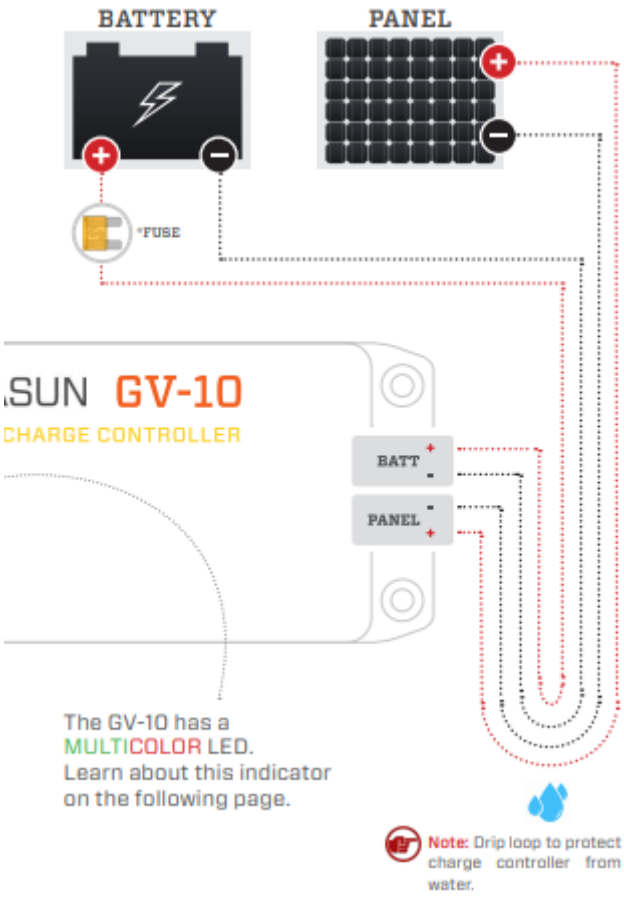


Fig. 19: Conexões do controlador de carga - Fonte: Genasun GV-10 Manual

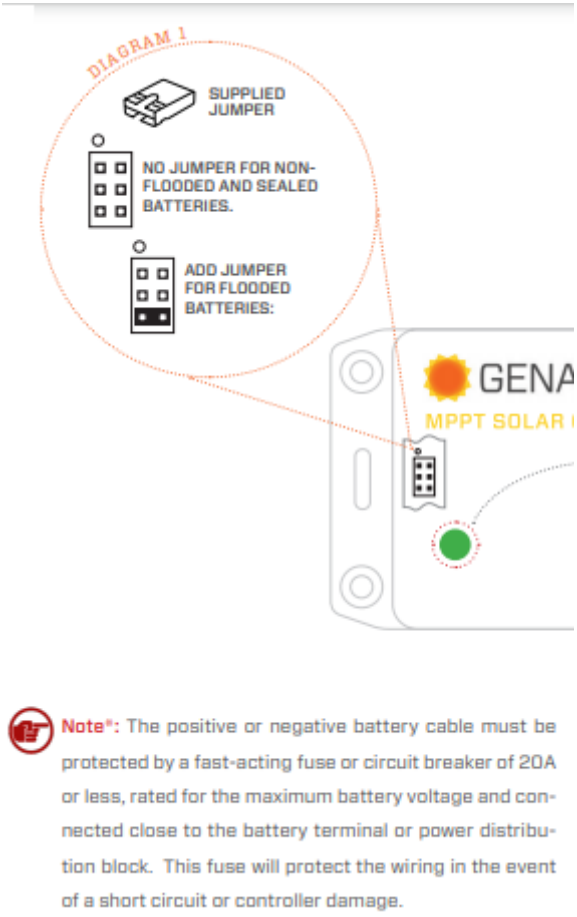


Fig. 20: Seleção de modos do controlador de carga - Fonte: Genasun GV-10 Manual

Specifications:

	GV-10-Pb-12V		GV-10-Li-**.*V	
Maximum Recommended Panel Power:	140W		GV-10-Li-12.5V	120W
			GV-10-Li-14.2V	140W
			GV-10-Li-16.7V	160W
Rated Battery (Output) Current:	10.5A		10.5A	
Nominal Battery Voltage:	12V		N/A	
Max Input Voltage:	34V		34V	
Recommended Max Voc at STC:	27V		27V	
Minimum Battery Voltage for Operation:	8.5V		8.5V	
Input Voltage Range:	0-34V		0-34V	
Maximum Input Short Circuit Current*:	10.5A		10.5A	
Maximum Input Current**:	19A		19A	
Charge Profile:	Multi-Stage with Temperature Compensation		CC-CV	
Charging Voltages:	FLOODED Setting	SEALED Setting		
Equalization Voltage:	15.0V	-	-	
Equalization Time:	2 Hours	-	-	
Equalization Interval:	30 Days	-	-	
Bulk Voltage:	14.6V	14.3V	-	
Absorption Voltage:	14.4V	14.1V	-	
Absorption Time:	2.5 Hours		-	
Float Voltage (Pb models) or CV Voltage (Li models):	13.5V	13.7V	GV-10-Li-12.5V	12.5V
			GV-10-Li-14.2V	14.2V
			GV-10-Li-16.7V	16.7V
Battery Temperature Compensation:	-28mV/°C (referred to 25°C)		disabled	
Operating Temperature:	-40°C - 85°C			
Maximum Full Power Ambient:	70°C			
Electrical Efficiency:	96% - 98% typical			
Tracking Efficiency:	99+% typical			
MPPT Tracking Speed:	15Hz			
Night Consumption:	0.9mA (900uA)			
Environmental Protection:	IP40, Nickel-Plated Brass & Stainless Hardware			
Connection:	4-position terminal block for 10-30AWG wire			
Certifications:	cETLus Safety, Recognized Component cETLus HazLoc (C1D2), CE, FCC, RoHS			
Weight:	6.5oz., 185g			
Dimensions:	5.5x2.5x1.2", 14x6.5x3.1cm			
Warranty:	5 years			

*Panel Isc. Maximum input power and maximum input voltage requirements must also be respected.
**Maximum current that the controller could draw from an unlimited source. This specification is not intended for determining PV input.

Fig. 21: Especificações controlador de carga - Fonte: Datasheet Genasun GV-10

Referências

- *Genasun GV-10 Lithium 16.7 Volt MPPT* - https://genasun.eu/products/genasun-gv-10-lithium-16-7-volt-mppt

3DRPOWER

O 3DRPOWER é um sistema de monitoramento de energia de bateria. Ele é ligado ao PixHawk e permite que o piloto automático posso executar alguma rotina coveniente quando a bateria apresentar nível de carga crítico.

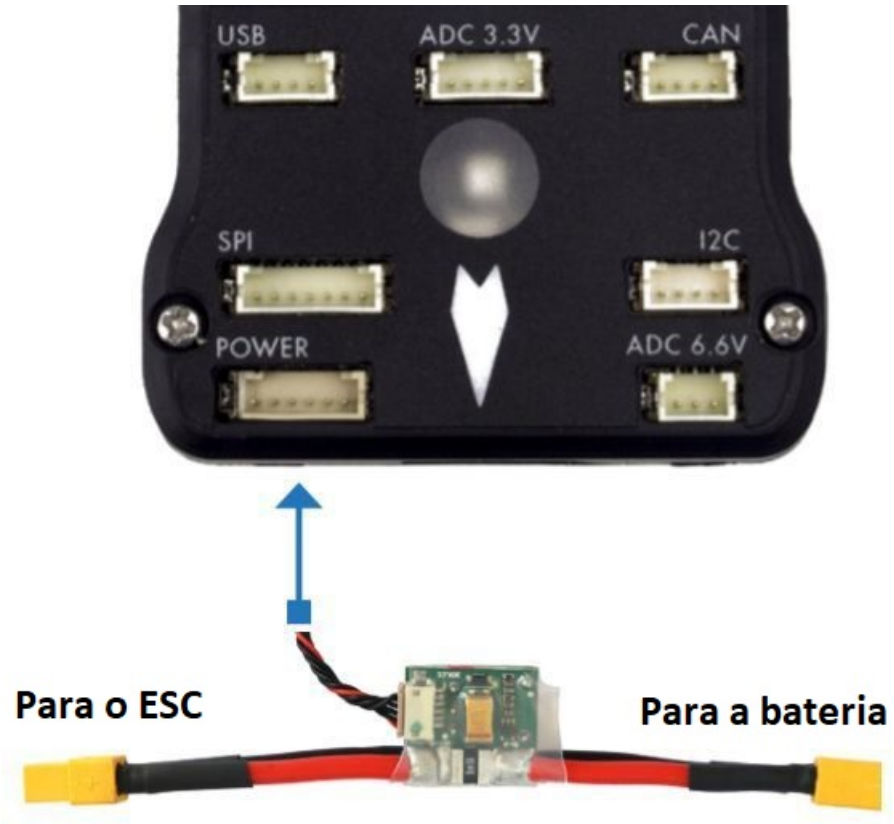


Fig. 22: Módulo 3DRPOWER

Configurações dos pinos:

- BATT_MONITOR: 3 para medir apenas a tenstão e 4 para medir a tensão e a corrente (necessário reiniciar a placa);
- BATT_VOLT_PIN: 2. Pino de conexão do piloto automátio;
- BATT_VOLT_MULT: Conversão de tensão analógica no pino para a tensão da bateria;
- BATT_CURR_PIN: 3 para o pino do piloto automático conectado ao módulo de energia;
- BATT_AMP_PERVLT: Conversão de tensão analógica no pino para a corrente da bateria;
- BATT_AMP_OFFSET: valor de tensão no pino de corrente do módulo quando não corrente advinda da bateria;

Referências

- *Common Power Module* - https://ardupilot.org/copter/docs/common-3dr-power-module.html

Bateria

A bateria utilizado no VANT Solar é do modelo (Li-Po) Turnigy 5000mAh C20 de íon de polímero.

Fig. 23: Turnigy 5000mAh 3S 20C LiPo Pack w/XT-60 - Fonte: Hobbyking

- Capacidade: 5000.00 mAh;
- Máxima taxa de carga: 2.00 C;
- Descarga: 20.00 c;
- Altura: 51 mm;
- Comprimento: 143.00 mm;
- Largura: 23.00 mm;

Referências

- *Turnigy 5000mAh 3S 20C LiPo Pack w/XT-60* - https://hobbyking.com/pt_pt/turnigy-5000mah-3s-20c-lipo-pack-xt-90.html?___store=pt_pt

ESC (Electronic Speed Control)

Regulador de controld de velocidade de um motor de corrente CC brushless. O modelo utilizado no VANT com painéis solares é o Volantex R/C Super Decathlon 40A Brushless ESC.

- Input: 2-4 Cells Lipo ou 15-12 cell Nimh;
- Compativo com BEC (Battery Ellimination Circuit) de 5V/4A;

Referências

- *Volantex R/C Super Decathlon 40A Brushless ESC* - https://www.amainhobbies.com/volantex-r-c-super-decathlon-40a-brushless-esc-vlt-747516/p416882
- "Volantex RC 40A Pro Switch-Mode BEC Brushless ESC" - https://hobbystation.co.nz/volantex-rc-40a-pro-switch-mode-bec-brushless-esc/

PIXHAWK

A imagem abaixo apresenta as conexões dos sensores e demais itens inclusos no Pixhawk. Cada parte será analisada com mais detalhes nas seções a seguir.

Fig. 24: Volantex R/C Super Decathlon 40A Brushless ESC - Fonte: Amainhobbies

Campainha e interruptor de segurança

A campainha fornece sinais sonoros que indicam a situação do VANT. Enquanto o interruptor atua na segurança da aeronave, bloqueando e desbloqueando os motores.

Nota: O interruptor de segurança é ativado por padrão e quando ativado, não permite o voo, bloqueando os motores. Para desativar o modo de segurança, pressione e segure o interruptor por 1 segundo. Você pode ativar o modo de segurança novamente pressionando o interruptor.

Para conectar a campainha e o interruptor de segurança (itens obrigatórios), basta liga-lós ao Pixhawk como mostrado abaixo.

Divisor I2C

O *slitter* I2C expande a quantidade de portas I2C permitindo a conexão de até quatro periféricos ao Piwhawk. Utilize um cabo de 4 fios para conectar o *slitter* I2C e para alimentar uma bússola externa, um display LED, um sensor de velocidade do ar digital e/ou qualquer outro periférico compatível ao veículo.

Sensor de velocidade do ar

Em edição...

GPS + Compass

O GPS, outro dispositivo obrigatório, deve ser conectado à porta GPS (6 pinos) usando o cabo de 6 fios fornecidos no kit. A conexão da bússola é opcional, porém recomendamos fortemente sua utilização. Para conecta-lá, ligue um cabo de 4 fios a uma porta I2C do *slitter* I2C, como mostrado abaixo.

Nota: O GPS/bússola deve ser montado no chassi da aeronave o mais longe possível de outros componentes eletrônicos, com a seta indicadora voltada para a frente e o mais alinhada possível com o Pixhawk.

Rádio controle

O sistema de rádio controle (RC) é necessário caso deseje controlar manualmente seu veículo, dado que o Pixhawk não requer um sistema de rádio para modos de voo autônomo.

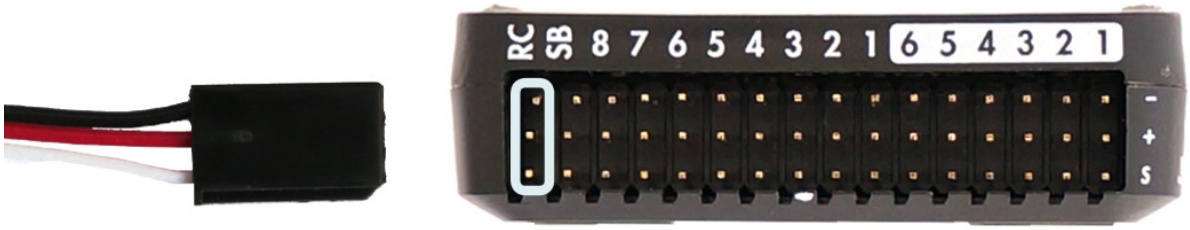
Para conectar o sistema de rádio controle, será necessário selecionar um transmissor/receptor compatível e depois vinculá-lo para que eles se comuniquem.

Dica: Leia as instruções que acompanham seu transmissor/receptor.

As instruções a seguir mostram como conectar os diferentes tipos de receptores ao Pixhawk:

- Os receptores Spektrum e DSM se conectam à entrada SPKT/DSM .

- Os receptores PPM-SUM e S.BUS conectam-se aos pinos de terra, potência e sinal RC, conforme mostrado.

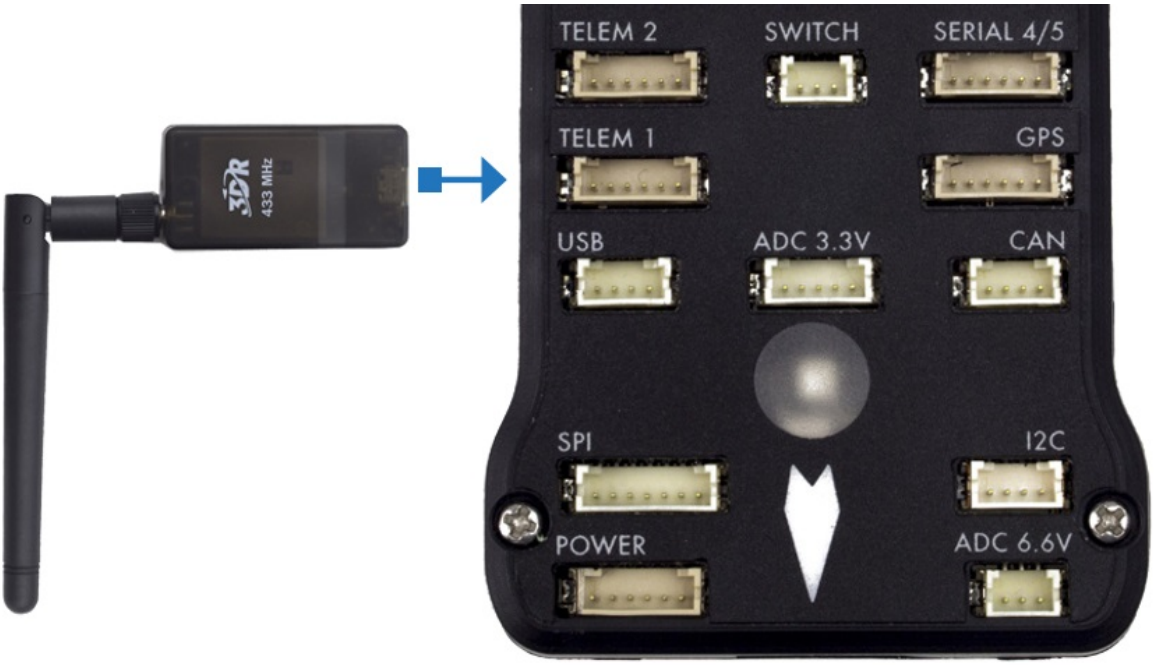


- Os receptores PPM e PWM que possuem um fio individual para cada canal devem se conectar à porta RC por meio de um codificador PPM (os receptores PPM-Sum usam um único fio de sinal para todos os canais).

Para obter mais informações sobre a seleção de um sistema de rádio, a compatibilidade do receptor e a ligação do seu par transmissor e receptor, consulte: Transmissores e receptores de controle remoto.

Telemetria

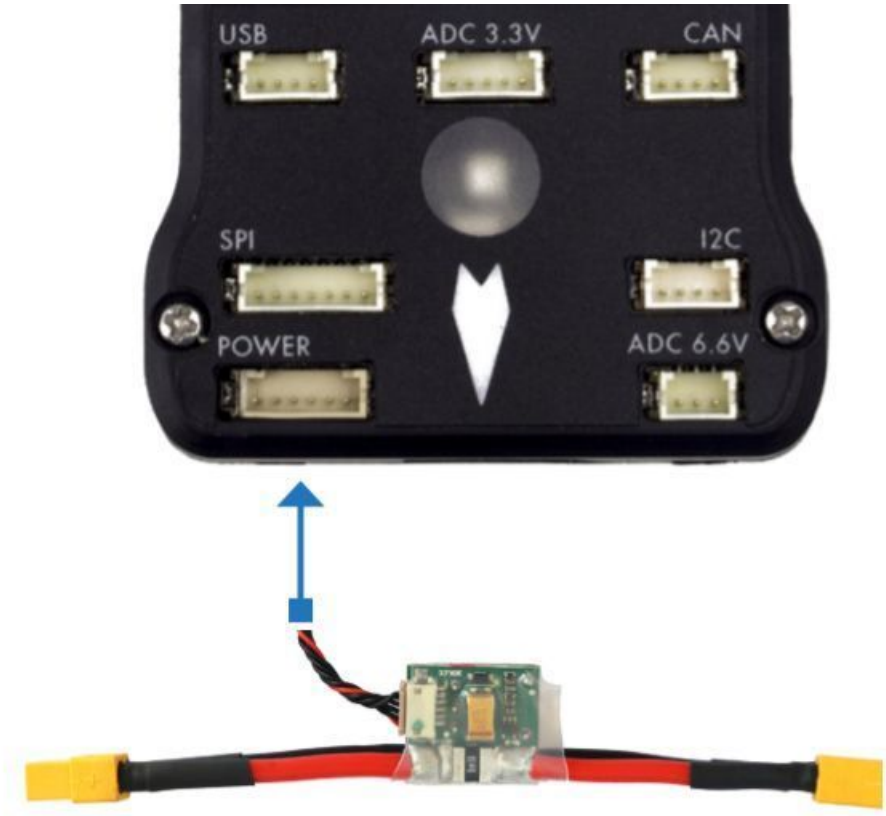
Os modens de telemetria podem ser usados para comunicar e controlar um veículo em voo a partir de uma estação terrestre (por exemplo, você pode direcionar o VANT para uma posição específica ou carregar uma nova missão). Um modem deve ser conectado ao seu veículo, como mostrado abaixo. O outro modem deverá ser conectado ao computador da estação terrestre ou dispositivo móvel (geralmente por uma porta USB).



Módulo de energia

O módulo de energia (*Power module* - PM) fornece energia ao controlador de voo da bateria e também envia informações sobre a corrente analógica e a tensão fornecida pelo módulo (incluindo a energia do controlador de voo e dos motores, etc.).

A saída do modulo de energia (PM) deve ser conectada à porta **POWER** do Pixhawk usando um cabo de 6 fios, como apresentado na imagem. A entrada do modulo deverá ser conectada a uma bateria de LiPo, enquanto a saída principal será responsável por fornecer energia aos ESCs e motores da aeronave (possivelmente através de uma placa de distribuição de energia, a depender da aeronave).



Sensor de distancia

O Pixhawk suporta vários sensores de distância diferentes, incluindo os Lidars (que usam lasers ou raios infravermelhos para medições de distância) e Sonars (que utilizam som ultrassônico), e também incluem os buscadores de alcance LED Maxbotix Sonar e Pulsed Light. Dessa forma, a instalação varia de dispositivo para dispositivo. Mais informações a respeito da configuração dos sensores pode ser visualizada em Rangefinders.

Fig. 25: Exemplo de alguns sensores de distância compatíveis

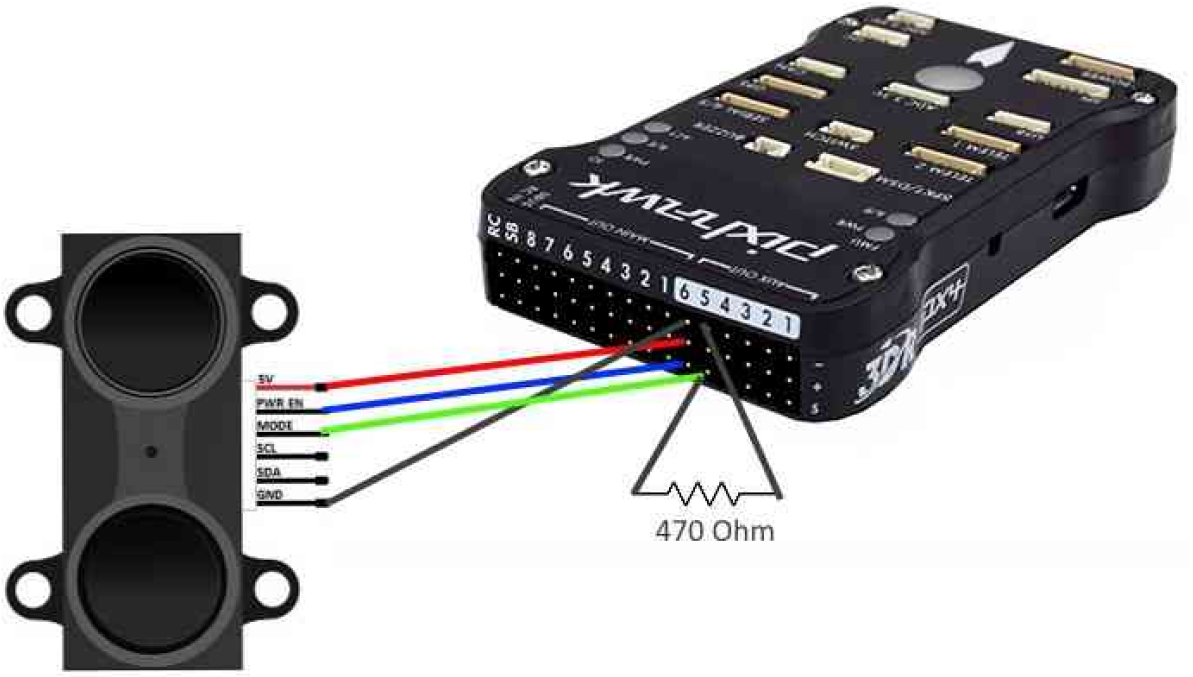
Para implementar o projeto, escolheu-se o sensor Lidar para habilitar a função de pouso automático devido sua maior precisão em relação aos demais. O sensor lidar pode ser conectado ao Pixhawk de duas formas, através do protocolo I2C na porta I2C (ou I2C *slitter*) ou por *pulse-width-modulation* (PWM) na trilha PWM.

De acordo com a documentação do Pixhawk, o lidar utilizado apresenta problemas de interferência com outros dispositivos quando conectado na porta I2C. Assim, escolheu-se a conexão por PWM. Um diagrama de conexão pode ser vista na tabela abaixo e o esquema de montagem pode ser visto na figura a seguir, onde o valor do resistor pode variar entre 200Ω e 1kΩ[1].

[1] Eduardo Moura Cirilo Rocha. 2017. Desenvolvimento de um sistema com veículos aéreos não-tripulados autônomos, Universidade de Brasília, Brasil

Table 3: Diagrama de conexão entre o Lidar e o Pixhawk

Sinal LIDAR-Lite	Sinal Pixhawk
J1	CH6 Out - V+
J2	CH6 Out - Signal (sinal interno 55)
J3	CH5 Out - Signal (sinal interno 54)
J4	
J5	
J6	Ch6 Out - Ground



Mais detalhes sobre a conexão podem ser encontrados em LIDAR-Lite Rangefinder.

Mais informações e referências

- Pixhawk Wiring Quick Start - PX4 User Guide
- Basic Assembly - PX4 User Guide
- Pixhawk Series - PX4 User Guide
- Peripheral Hardware - Ardupilot Docs

SERVOS

Especificações:

- Product Dimensions : 15.08 x 10 x 1.27 cm; 14.51 Grams
- Date First Available : 21 August 2017
- Manufacturer : HRP Distributing, Inc.
- ASIN : B016N411HC
- Item model number : VLX757323
- Manufacturer : HRP Distributing, Inc.

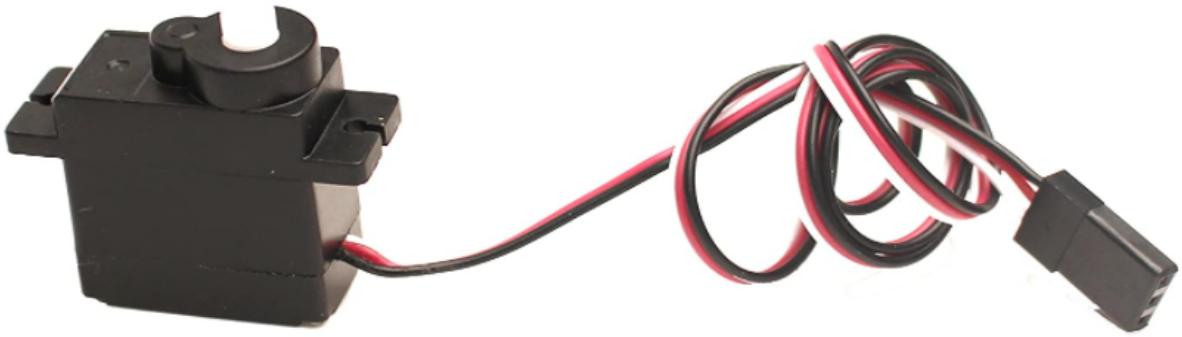


Fig. 26: Volantex RC 757323 Ranger EX Servo 9G Metal Gear- Fonte: Amazon

- Item Weight : 14.5 g
- Item Dimensions LxWxH : 15.1 x 10 x 1.3 Centimeters

Referências

- Volantex RC 757323 Ranger EX Servo 9G Metal Gear - https://www.amazon.in/Volantex-757323-Ranger-Servo-Metal/dp/B016N411HC

Motor

Fig. 27: Turnigy Aerodrive SK3 - 3536-1400kv Brushless Outrunner Motor - Fonte: Hobbyking

Especificações:

- Acontece: 12T;
- Voltagem: 3 ~ 4S Lipoly;
- RPM / V: 1.400 kv;
- Resistência interna: 0,021 Ohm;
- Max Loading: 40A;
- Max Power: 590W;
- Shaft Dia: 5,0 milímetros;
- Furos de parafuso: 25mm;
- Rosca do parafuso: M3;
- Peso: 110 g;
- Motor Plug: 3,5 milímetros conector de bala;

Referências

´Turnigy Aerodrive SK3 - 3536-1400kv Brushless Outrunner Motor´ - https://hobbyking.com/pt_pt/turnigy-aerodrive-sk3-3536-1400kv-brushless-outrunner-motor.html

FPV (First Person View)

Fig. 28: FPV

Modelo: Fatshark 700TVL High Resolution FPV Tuned CCD Camera V2 (NTSC) - Fonte: Hobbyking

Características:

- FPV tuned white balance and gain control
- 700 TVL high resolution camera
- Compact size

Especificações:

- Type: Colour
- Signal System: NTSC
- Pick Up Device: 1/3” SONY Super-HAD CCD
- Pixel: 726X 582
- Resolution: 700 TV Lines
- Min Illumination: 0.08 Lux /F1.2
- Scanning System: 2:1 Interlace
- Sync System: Internal
- S/N Ratio: More than 55dB
- Gain Control: Auto (FPV tuned)
- White Balance : Auto (FPV tuned)
- Lens: 3.6mm (no IR)
- Video output: 1.0V p-p (75Ω BNC)
- Power Supply: 3.3 – 5V DC
- Power Consumption: 110mA
- Dimensions: 27 X 25 X 13 mm (+14mm lens extrude)
- Weight : 31g
- Operating Temp: -10 to 50°C

Referências

- Fatshark 700TVL High Resolution FPV Tuned CCD Camera V2 (NTSC) - https://hobbyking.com/en_us/fatshark-700tvl-high-resolution-fpv-tuned-ccd-camera-v2-ntsc.html

1.2.3 Sistema obtenção e Gerenciamento de Energia

Nota: Este topico constém trechos replicados de artigos desenvolvidos no Laboratório de Robótica Aérea - UnB. Os artigos são referenciados no fim deste tópico.

Esse sistema é composto por todos os dispositivos que se relacionam diretamente com a conversão e manutenção da energia solar em energia elétrica. Toda a energia proveniente das células solares é ordenada, formando um fluxo contínuo de elétrons, sendo portanto de corrente contínua (CC). O sistema fotovoltaico pode ser classificado em «conectado à rede»ou «isolado (ou autônomo)».

Os sistemas conectados à rede elétrica requerem participação da concessionária de energia, enquanto os sistemas isolados, também chamados de off-grids, em geral utilizam baterias. Para finalidade dos VANTs, é adotado o sistema do tipo isolado, o qual é composto basicamente pelos itens abaixo:

- Painel solar;
- Controlador de carga;
- Inversor de corrente off-grid (utilizado apenas quando se deseja alimentar componentes de corrente alternada);
- Baterias.

Para conversão da luz solar em eletricidade é necessária a utilização de módulos fotovoltaicos compostos por células solares, conforme figura 2, em geral produzidas a partir de silício cristalino. O painel solar será composto pela associação em série das células solares, com a finalidade de converter a luz solar em energia elétrica.

O uso de células fotovoltaicas, formando um painel solar na superfície da asa de um VANT pode ser, portanto, um artifício para melhora de autonomia nesses veículos. Entretanto, para que a energia proveniente da luz solar seja convertida e entregue de forma ordenada e eficiente aos subsistemas que compõem ao VANT é necessário uma integração do sistema de gerenciamento de energia solar ao sistema de navegação e controle. As células solares utilizadas no projeto são como a mostrada na figura 2 e têm as especificações mostradas na tabela 1, onde Vmpp e V oc são as tensões de máxima potência e circuito aberto, respectivamente, e, Impp e Isc as correntes de máxima potência e de curto circuito, respectivamente:

Table 4: Especificações da célula Maxeon C60

Características	Valor
Dimensões	125 x 125 mm
Eficiência	21,8%
Vmpp	0,574V
Voc	0,682V
Impp	5,83A
Isc	6,24A

A função dos controladores de carga (ou sistema de gerenciamento de energia) em um sistema fotovoltaico está relacionada com a carga e descarga das baterias, evitando que operem em 4 níveis críticos, mas garantindo que a energia gerada pelo painel seja armazenada nas baterias com maior eficácia.

Os controladores de carga exercem papel fundamental em um sistema fotovoltaico, regulando as cargas e descargas excessivas. Dois principais tipos de tecnologia podem ser empregadas nesse tipo de sistema: PWM (pulse width modulation) e MPPT (maximum power point tracking). Os controladores PWM, mais convencionais, regulam a tensão da bateria por meio da variação da largura dos pulsos de corrente que são fornecidos a ela. Com isso, conforme as baterias se aproximam da carga total, o controlador é capaz de diminuir lentamente a energia aplicada, diminuindo as sobrecargas e prolongando a vida útil das baterias.

Por outro lado, os controladores dotados de algoritmo MPPT, possibilitam rastreamento de ponto de potência máxima ajustando continuamente a impedância percebida para que o sistema fotovoltaico opere no ou próximo ao ponto de máxima potência do painel fotovoltaico mesmo sob variação da irradiação solar, temperatura etc. Esses controladores são capazes de converter o excesso de tensão fornecida pelo painel fotovoltaico em aumento na corrente numa razão de mesma proporção, mantendo dessa forma a mesma potência fornecida. Ao receber 18V das células solares para alimentação da bateria de 12V, o controlador de carga regula essa tensão, de 18V para 12V, de forma a aumentar a corrente na mesma proporção, mantendo a mesma potência. Essa funcionalidade não é observada nos controladores PWM, isto é, qualquer tensão fornecida que excede a alimentação das baterias é desperdiçada.

As baterias de um sistema fotovoltaico devem ser compatíveis com o tipo de finalidade do sistema, isto é, alimentar a carga desejada. Baterias automotivas por exemplo não são desejáveis para essa finalidade uma vez que sua função principal é auxiliar a partida do veículo, fornecendo uma diferença de potencial em um pequeno período de tempo. As baterias utilizadas nesse estudo serão de íon de polímero (Li-Po) Turnigy 5000mAh C20.

Dimensionamento do painel solar

Nota: Este topico constém trechos replicados de artigos desenvolvidos no Laboratório de Robótica Aérea - UnB. Os artigos são referenciados no fim deste tópico.

Para determinação da quantidade de células solares, foi tomado como base o consumo energético dos subsistemas que compõem o VANT. O consumo de um motor de um VANT representa grande parcela no consumo total.

Para o modelo em análise, foi utilizado um motor de corrente contínua sem escovas Turnigy 3536 1400 kv, cuja potência nominal em rotação máxima é de 590 W.

Embora a potência máxima seja alta para que seja suprida por células solares em uma asa de avião, é importante ressaltar que, com base em observações experimentais, para decolagem do Volantex RC Ranger foi necessário apenas cerca de metade da potência do motor.

Em vôo reto e nivelado, foi observado que cerca de 35% a 40% da potência máxima do motor eram necessários para manutenção do vôo. Com base nisso, o sistema de energia solar que atendesse parcialmente essa potência seria satisfatório.

Para a potência de 236 W (40% de potência do motor) a ser suprida durante vôo de cruzeiro, pode-se calcular a potência do painel solar, considerando as perdas e irreversibilidades das conexões, sujeiras e fatores como variação de incidência solar, a partir da equação 1. A potência necessária do painel estimada é de 296 Wp 1, considerando as variações de Horas de Sol Pleno (HSP) para o estado de Brasília - DF [2]. A equação que determina a potência em Wp é expressa por:

Equação que determina a potência (Pm) em Wp:

$$Pm = \frac{L_i}{HSP.Red1.Red2}$$

Pm é a potência total necessária do painel, Li é o produto da potência nominal consumida pela quantidade de horas em operação. HSP é a quantidade de horas de sol pleno, com irradiação 1.000 W/m2 (para Brasília usa-se o pior caso de 4,72). Red1 e Red2 são fatores de irreversibilidade de temperatura, conexões em cabos, sujeira, incompatibilidades elétricas e etc, por padrão são adotados 0,75 e 0,9 respectivamente [2].

Referências

- GOMES, D. H. Configuração e estudo do sistema de gerenciamento de energia, navegação e controle como parâmetros ótimos na montagem e desempenho de um VANT com painéis solares. Faculdade de Tecnologia, Universidade de Brasília, 2019.

Associação das células

Nota: Este topico constém trechos replicados de artigos desenvolvidos no Laboratório de Robótica Aérea - UnB. Os artigos são referenciados no fim deste tópico.

As células fotovoltaicas podem ser associadas umas as outras em série ou em paralelo. É interessante trabalhar com baixas correntes de forma a evitar, em casos de curto circuito, que os sistemas sejam danificados. Além disso, os controladores de carga possuem um limite mínimo de tensão da bateria para que continuem operando, ou seja, é importante assegurar que a bateria esteja em níveis de tensão suficientemente altos.

Com os dois lados da asa, foi possível fazer a ligação de dois módulos em paralelo, beneficiado pela própria configuração da asa, ou de uma ligação única em sentido "circular" associando as células em série. Optou-se pelo segundo caso.

O posicionamento das células em série também é justificado posteriormente pelo algoritmo MPPT do controlador de carga, que é responsável por buscar o ponto ótimo na curva tensão-corrente, fornecendo sempre a mesma potência, independentemente do excesso de tensão fornecido.

As células foram soldadas com estanho umas as outras com conectores do tipo dog bone, que acompanham as células.

O ponto ótimo para potência fornecida se encontra, na curva azul devido as condições de operação, nas proximidades de 0,5V e 6A, representando uma geração ótima de aproximadamente 3W por célula, totalizando 108W.

A nova asa construída permite, nessa configuração, instalação de 36 células simetricamente dispostas com relação ao perfil central, associadas em série fornecendo uma diferença de potencial de 18V. A nova configuração representa uma ocupação de cerca de 70% da área de asa, conforme figura 6, e um aumento de 50% de células em relação à asa original da aeronave.

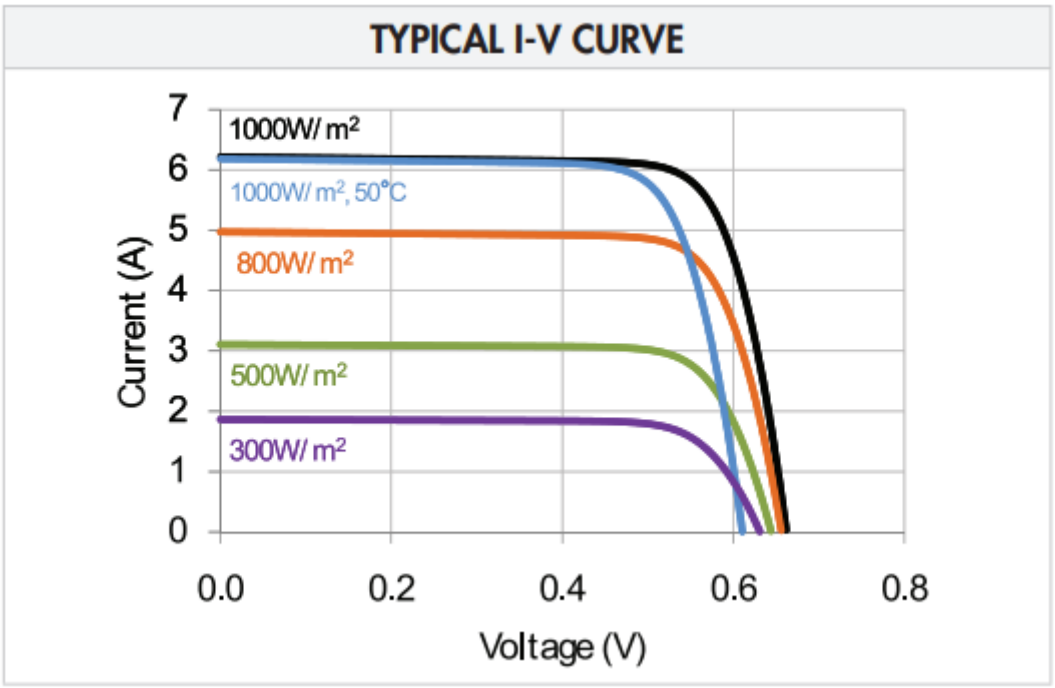


Fig. 29: Típica I-V curva - Fonte: Datasheet SUNPOWER C60 SOLAR CELL

Referências

- GOMES, D. H. Configuração e estudo do sistema de gerenciamento de energia, navegação e controle como parâmetros ótimos na montagem e desempenho de um VANT com painéis solares. Faculdade de Tecnologia, Universidade de Brasília, 2019.

1.2.4 Projeto Aerodinâmico

O Projeto Aerodinâmico do VANT Solar é um híbrido entre um aeromodelo já existente, o Ranger EX Volatex RC, e o projeto e implementação de uma nova asa que suprisse o aspecto de área disponível para a instalação da células solares.

Foi realizado primeiramente a escolha do perfil de acordo com as as carecterísticas de vôo da aeronave e foi realizado posteriormente a simulação da asa no software XFRL5.

Por fim realizou-se a construção da asa e sua instalação no aeromodelo. Os detalhes de cada estapa estão descritas nos tópicos a seguir.

Aeromodelo

O modelo do avião utilizado para executar o voo é o Ranger EX 757-3 da marca Volantex RC, mostrado na Figura 1. O modelo é um avião monomotor de asa alta, com trem de pouso convencional, ou seja, a bequilha é fixada após o leme do avião.

Especificações técnicas:

- Wings : 1980mm (77.9 in)
- Overall Lenght: 1170mm (46 in)
- Flying weight: 1500g
- Propeller: 1060 Propeller

Fig. 30: Aeronave Ranger EX Volatex RC

- Motor: 3715 powerful out runner brushless motor
- Speed controller: Easy-Plug 40A Switch-mode BEC brushless ESC
- Servo 9-gram Servo * 6pcs
- Recomended battery: 11.1V/14.8V 1800mAh ~ 10000mAh Li-Po
- Radio: 2.4Ghz 6-Channels
- Battery Charger: Not included
- Ailerons: Yes
- Flaps: Yes
- Minimum Age Recomendations: 14 years
- Experience Level: Beginer ~ Expert
- Rekomended Environment: Outdoor
- Assemlby Time: Less than 30 minutes
- Is Assembly Required: Yes

Referências

- PEREIRA, G. H. S. Análise de parâmetros aerodinâmicos e configuração de sistemas de comunicação e captação de imagem para um VANT com painéis solares nas asas. Faculdade de Tecnologia, Universidade de Brasília, 2019.
- Volantex RC Ranger EX Long Range FPV / UAV platform Unibody big weight carrier (V757-3) PNP: https://www.volantexrc.eu/volantex-rc-ranger-ex-long-range-fpv-uav-platform-unibody-big-weight-carrier-v757-3-pnp-p-224.html

Escolha do Perfil

Nota: Este topico constém trechos replicados de artigos desenvolvidos no Laboratório de Robótica Aérea - UnB. Os artigos são referenciados no fim deste tópico.

O perfil escolhido para a asa do VANT solar deve atender todos os requisitos propostos no projeto. Um deles é poder fixar duas fileiras de paneis solares quadrados (12cm×12cm) em sua superfície superior, além de alocar os dispositivos hipersustentadores. Considerando as células como placas finas e extremamente frágeis quanto às tensões de flexão, o perfil escolhido deve ter pouca curvatura para evitar que as células sejam danificadas quando ajustadas ao formato do perfil.

Sendo o valor de corda o comprimento entre o bordo de ataque e o bordo de fuga, através de experimentos chegou-se à uma dimensão de 32cm para a corda do perfil escolhido. Esse tamanho permite que todas as células sejam alocadas com segurança e ainda ter espaços para os servos e as superfícies hipersustentadoras.

Assim, considerando o formato do aerofólio com as restrições apresentadas acima, além de seu uso generalizado no meio do aeromodelismo, escolheu-se o perfil CLARK-Y 11.7 % smoothed. Os pontos, no eixo cartesiano, foram obtidos a partir da base de dados do Departamento de Engenharia Aeroespacial da University of Illinois at Urbana-Champaign1 e usados para a análise numérica (ver seção 3.3) e para a construção do CAD (do inglês Computer Aided Design) no software de modelagem CATIA V5R21, que serão apresentados na seção «Simulações Numéricas no XFRL5».

Referencias

- PEREIRA, G. H. S. Análise de parâmetros aerodinâmicos e configuração de sistemas de comunicação e captação de imagem para um VANT com painéis solares nas asas. Faculdade de Tecnologia, Universidade de Brasília, 2019.

Parâmetros Aerodinâmicos para o Perfil

Nota: Este topico constém trechos replicados de artigos desenvolvidos no Laboratório de Robótica Aérea - UnB. Os artigos são referenciados no fim deste tópico.

A Figura a seguir apresenta a variação do Cl pelo ângulo de ataque, α. No detalhe dentro da imagem, é possível observar a mesma curva ampliada para facilitar a identificação do ângulo de máximo.

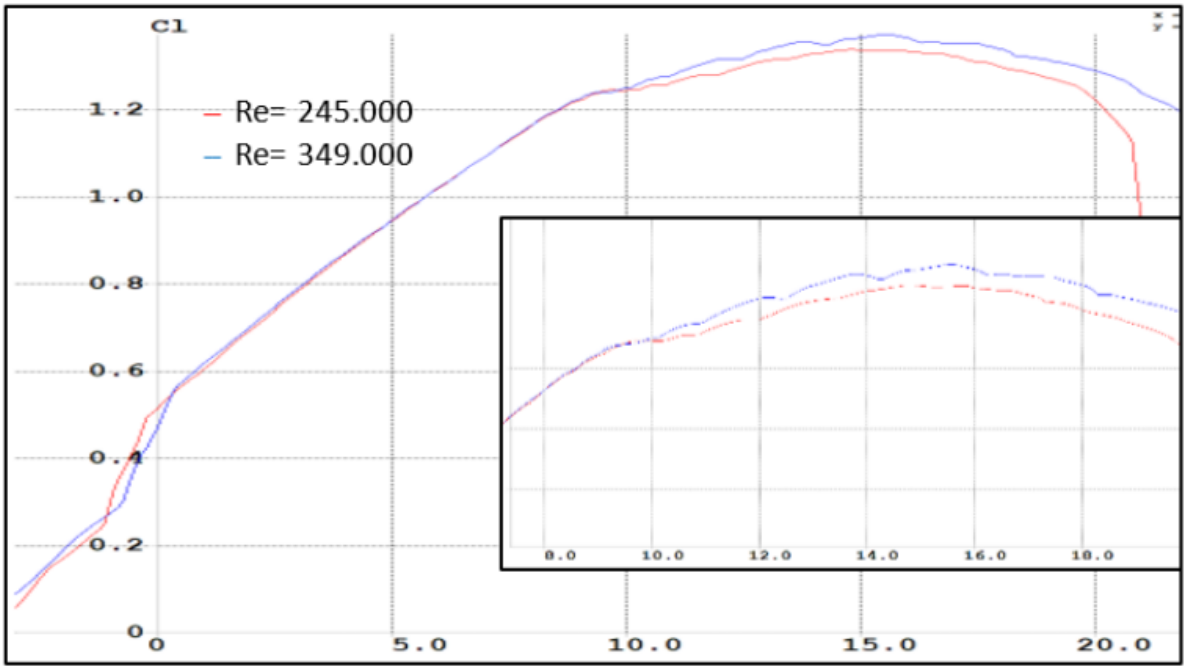


Fig. 31: Cl pelo ângulo de ataque

A curva presente na Figura 6 nos dá a informação sobre o ângulo onde o Cl é máximo, isso é, aumentando o ângulo a partir desse valor, o aeromodelo entra em stall, que é a situação na qual há perda de sustentação. Avaliando parte do gráfico ampliada, temos que o ângulo de stall é αstall = 15$\circ$ para Re = 245.000.

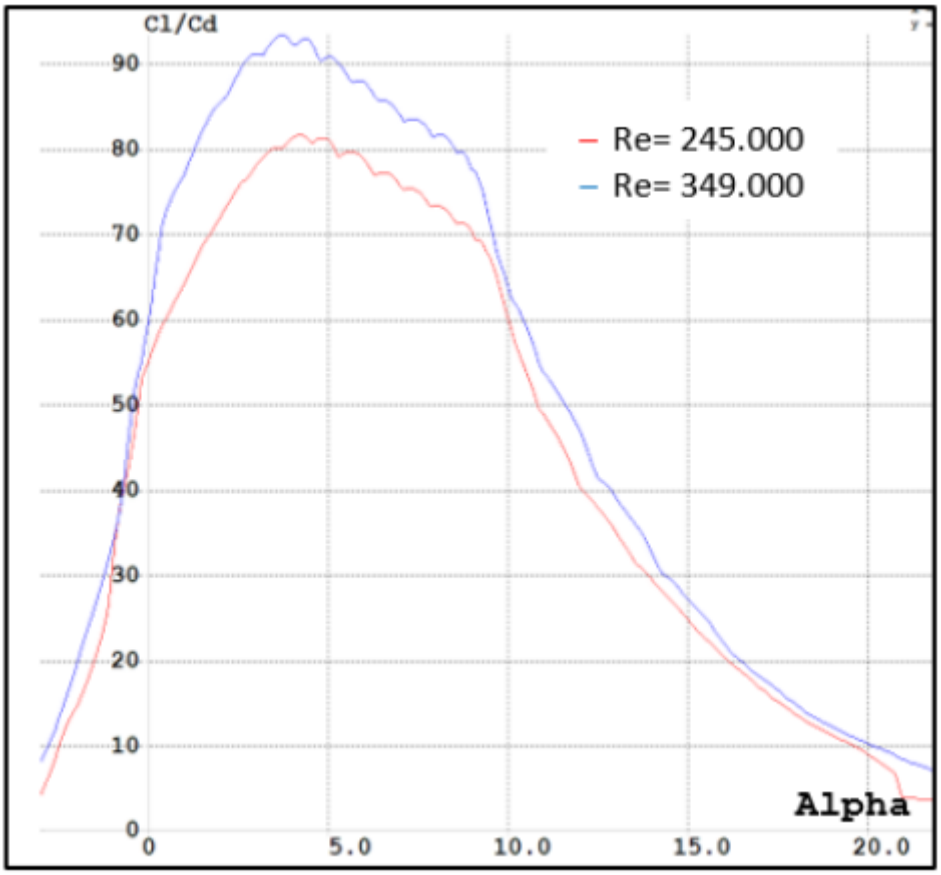


Fig. 32: Razão Cl/Cd pelo ângulo de ataque

O objetivo durante o procedimento de decolagem é que o ângulo de subida não se aproxime do ângulo de stall. Mediu-se, ainda com um nível eletrônico, o ângulo de ataque da asa e obteve-se 7 o valor de α = 5$\circ$, isso quando a fuselagem estava paralela a linha do chão. Essa informação, aliada à curva da Figura 6, permite dizer que o ângulo de ataque de decolagem está distante do de stall, trazendo mais segurança ao voo. Esses valores se assemelham às curvas presentes em (LYON, 2001) para testes com número de Reynolds aproximados para o perfil Clark-Y.

A Gráfico da razão Cl/Cd pelo ângulo de ataque apresenta outra curva na qual, no eixo das ordenadas, temos a razão Cl/Cd e no eixo das abcissas, o ângulo de ataque. Analisando o gráfico para Re = 349.000, obtido para voo nivelado, temos que o melhor ângulo de ataque para essa configuração é α = 4$\circ$. Avaliando o parâmetro Cl1,5/Cd, percebe-se que o ângulo de máxima autonomia está na região que varia α entre 4o a 6o . Região que abrange o ângulo de ataque em voo nivelado.

Referências

- PEREIRA, G. H. S. Análise de parâmetros aerodinâmicos e configuração de sistemas de comunicação e captação de imagem para um VANT com painéis solares nas asas. Faculdade de Tecnologia, Universidade de Brasília, 2019.

Comportamento Aerodinâmico da Asa

Nota: Este topico constém trechos replicados de artigos desenvolvidos no Laboratório de Robótica Aérea - UnB. Os artigos são referenciados no fim deste tópico.

A simulação numérica da asa abordou dois momentos crítico para o voo: a decolagem e o voo horizontal. Os resultados das simulações dos perfis permitiram avaliar o comportamento da asa em determinadas velocidade e atitude, que serão expostos nessa seção.

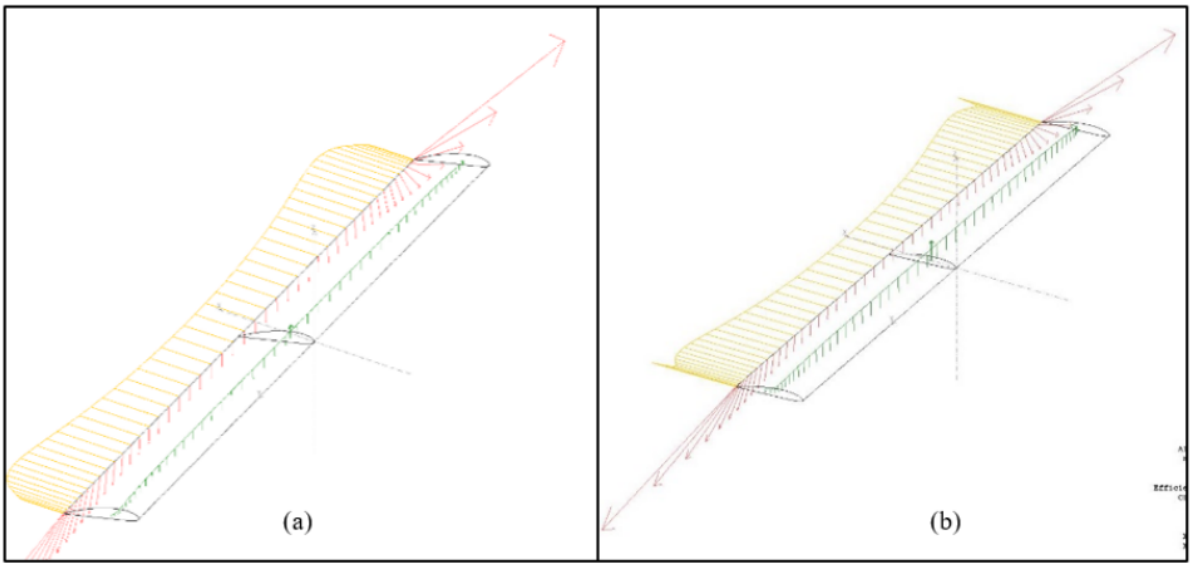


Fig. 33: Simulação numérica da asa para operação de decolagem para (a) ângulo de ataque de 13o e velocidade de 6 m/s e (b) para ângulo de ataque de 5o a velocidade de 12 m/s

A Figura Simulação numérica da asa para operação de decolagem apresenta os três parâmetros (arrasto induzido, o efeito de downwash e a sustentação) durante a decolagem. Na imagem (a), é representado o momento quando a quilha do avião está tocando o solo, ou seja, um ângulo de ataque de 13o em uma velocidade escolhida de 6m/s. Já a imagem (b) da Figura 8, mostra o momento em que a quilha sai do chão, fazendo com que a fuselagem tenha 0o em relação ao solo, o que resulta em um α de 5o .

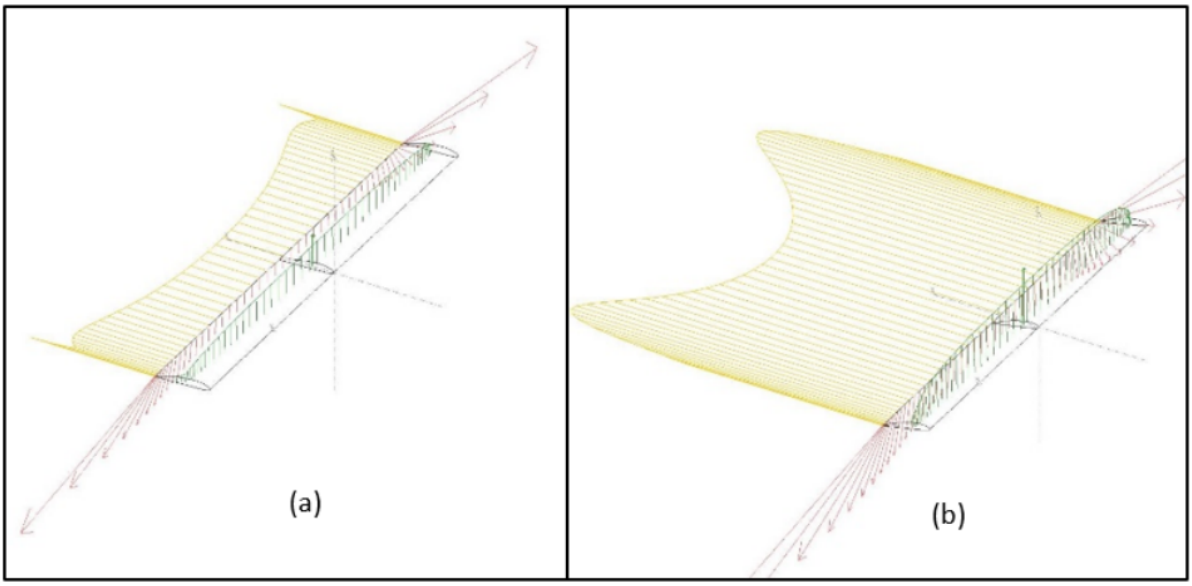


Fig. 34: Simulação numérica para Re = 349.000 do (a) ângulo de máxima razão Cl/Cd e para (b) ângulo de ataque de 12o

É perceptível que a curva amarela, representando o arrasto induzido, diminui quando a quilha do trem de pouso se levanta do chão, assim como também percebe-se que a sustentação, indicada pelos vetores em verde, estão maiores para $\alpha = 5\circ$. O downwash, como esperado, é maior nas pontas das asas, acrescentando mais arrasto neste local. Já em condição de voo horizontal com ângulo de ataque ótimo, $\alpha = 4\circ$, temos a máxima diferença entre o arrasto e a sustentação, como mostrado na Figura Simulação numérica para Re. É perceptível o aumento do arrasto quando analisamos ângulos de ataque maiores (por exemplo, $\alpha = 12\circ$) para a mesma velocidade e atitude de voo, como mostra a Figura em (b).

Construção da Asa

Nota: Este topico constém trechos replicados de artigos desenvolvidos no Laboratório de Robótica Aérea - UnB. Os artigos são referenciados no fim deste tópico.

Todas as peças foram cortadas a laser, para que haja uma perfeição nas medidas. A primeira peça projetada é a que tem o mesmo formato da abertura superior do avião. Ela servirá como ligação da fuselagem com a asa. A peça, denominada de 'Fixador Central' (com espessura de 10mm), tem suas dimensões apresentadas à esquerda da Figura, e à direta, a peça produzida.

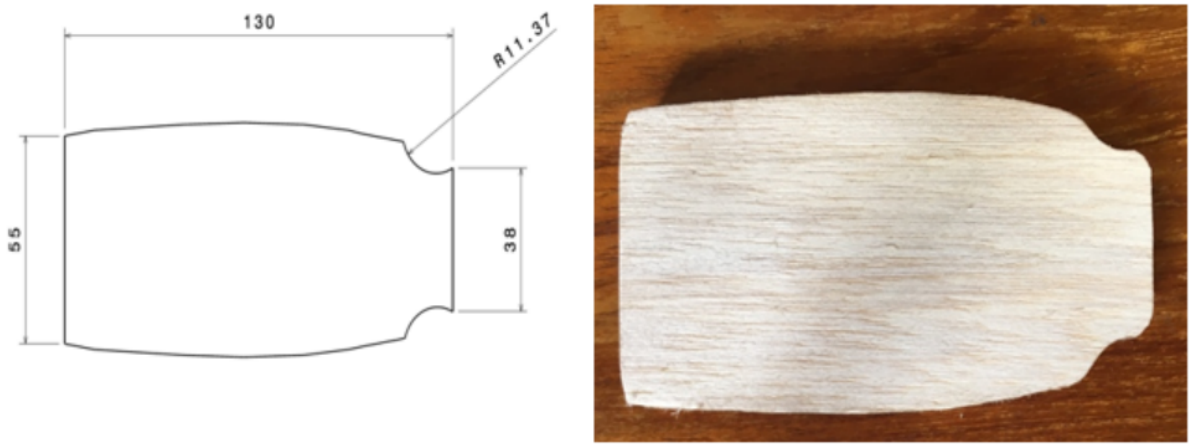


Fig. 35: Desenho esquemático do Fixador Central (Esquerda) e a peça produzida em madeira de balsa (direita)

Independente do perfil escolhido, existem vários formatos que podem ser adotados na escolha de uma asa. As mais comuns, usadas em grandes aeronaves, são as em formato delta, trapezoidal ou as elípticas. Todas elas requerem uma alta complexidade de construção. Levando em consideração a limitação do projeto, o formato escolhido foi a retangular, por ter uma fabricação mais simples comparadas às outras apresentadas anteriormente. Para dar rigidez e poder resistir aos esforços estruturais, existe uma estrutura de reforço chamado longarina, que é disposta ao longo do eixo transversal e serve de ligação entre os perfis. As longarinas do projeto foram fabricadas em fibra de carbono por serem resistentes e pelo seu baixo peso, comparado a outros metais. Isso permite que a asa não seja extremamente rígida, o que pode ocasionar um fratura abrupta da asa (MEGSON, 2016).

No projeto foram utilizadas longarinas de três diâmetros diferentes, sendo que as duas com maiores diâmetros se localizam na raiz da asa e têm 21,5 mm e 8 mm. As longarinas que se encontram mais perto da região da asa têm 5 mm de diâmetro. Essa diferença de diâmetros é de suma importância pois será o fator que determinará cada seção da asa. A asa foi dividida em cinco (5) seções diferentes, todas elas delimitadas pelas diferenças de diâmetros das longarinas. As três regiões possíveis em que a longarina pode estar é perto do bordo de ataque (denominada de 'Longarin B.A.'), perto do bordo de fuga (denominado de 'Longarina B.F.') ou entre essas duas, no qual chamaremos de 'Longarina Central'. As seções podem ser observadas na Figura 3.

As características de cada seção, com os diâmetros das longarinas, estão apresentadas na tabela Diâmetros das longarinas e comprimento das seções da asa. No caso de haver o símbolo '-' em alguma célula, indica que naquela seção não existe aquela longarina, já o símbolo '(2x)' representa uma quantidade de duas longarinas com determinado diâmetro, representado pela letra grega ϕ.

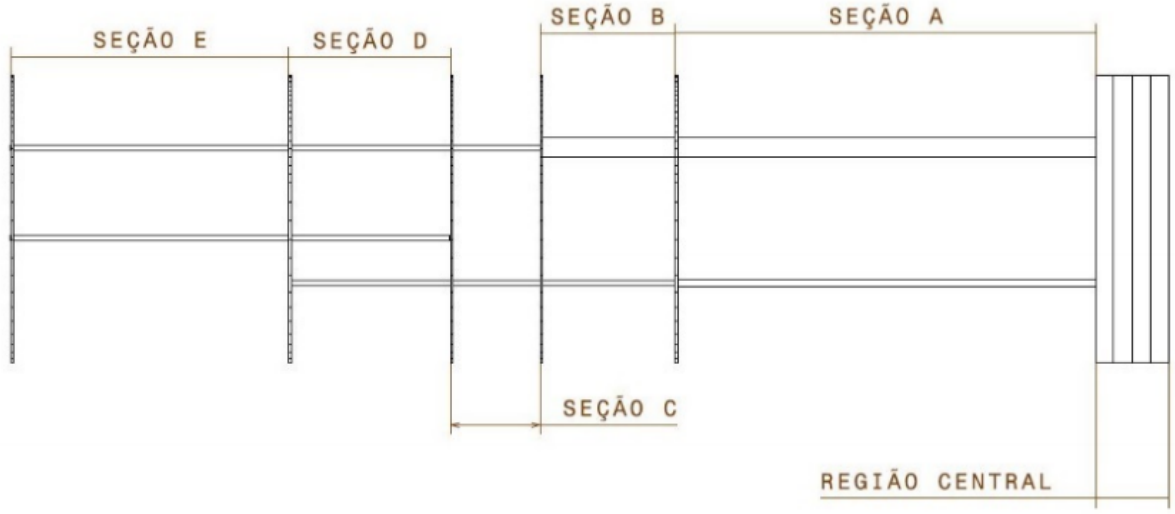


Fig. 36: Vista Superior das cinco seções delimitadoras da asa do VANT solar

Table 5: Diâmetros das longarinas e comprimento das seções da asa

Seção	Comprim. (mm)	Longarina B.A (mm)	Longarina Central (mm)	Longarina B.F. (mm)
A	470	21,5	•	8
B	150	21,5	•	5
C	100	5 (2x)	•	5
D	180	5 (2x)	5	5
E	310	5 (2x)	5	•

A tabela Diâmetros das longarinas e comprimento das seções da asa apresenta a quantidade de perfis necessários em cada seção e a distância que cada um deve ser colocado do outro. O layout interno de cada perfil segundo a sua seção se encontra no Anexo desse relatório.

Table 6: Especificações de cada perfil por seção

Perfil	Seção	Quantidade	Distância entre perfil (mm)
1	A	5	94
2	B	2	75
3	C	1	•
4	D	2	18
5	E	3	155

Com as características definidas, os perfis alocados, com as dimensões de suas respectivas 5 seções, podem ser observados na Figura 4. Lembrando que o esquema apresenta a semi-envergadura da asa, assim como o CAD apresentado na Figura 5.

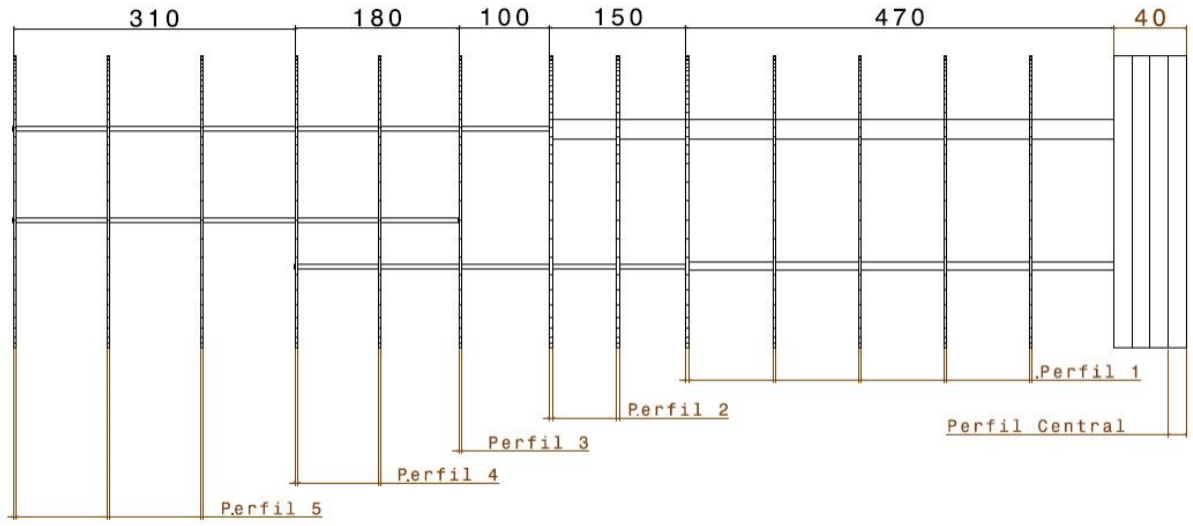


Fig. 37: Asa completa com as cotas e perfis correspondentes em cada seção.

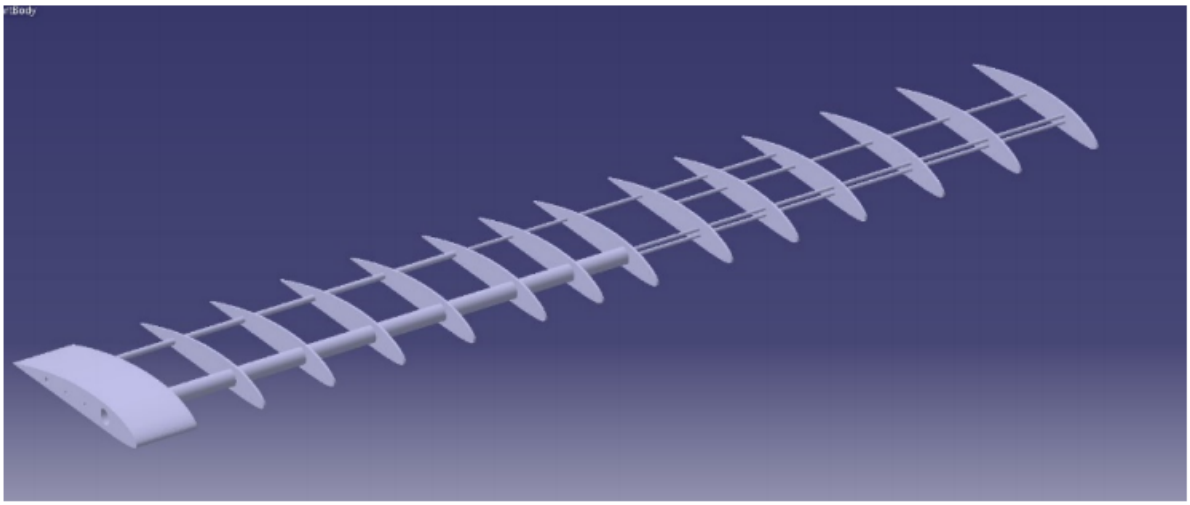


Fig. 38: Modelagem computacional no CATIA da metade da asa

Referências

- PEREIRA, G. H. S. Análise de parâmetros aerodinâmicos e configuração de sistemas de comunicação e captação de imagem para um VANT com painéis solares nas asas. Faculdade de Tecnologia, Universidade de Brasília, 2019.

Simulações Numéricas no XFRL5

Nota: Este topico constém trechos replicados de artigos desenvolvidos no Laboratório de Robótica Aérea - UnB. Os artigos são referenciados no fim deste tópico.

Usando os conjuntos de pontos do perfil Clark-Y smoothed e, com o auxílio do software XFRL5 V6.47, é possível obter, através de simulações numéricas, vários parâmetros aerodinâmicos pertinentes do perfil. Para a realização dessas simulações, além das coordenadas do perfil, é preciso inserir o número de Reynolds. O número de Reynolds, abreviado por Re, é um número adimensional considerado um dos parâmetros mais importantes para a mecânica dos fluídos e representa, fisicamente, a razão entre forças inerciais e forças de viscosidade (ANDERSON Jr, 2010).

O valor da viscosidade cinemática é tabelado para vários tipos de escoamento, inclusive o ar. Usando os valores tabelados em (CENGEL; GHAJAR, 2009), considerando a temperatura média de Brasília de 25◦C, obtivemos que o valor da viscosidade cinemática é ν = 1, 562 · 105 m2/s. O valor de l pode ser considerado como o valor da corda da asa estudada, nesse caso, l = 0, 32 m. Sabendo que a velocidade do fluído, para esse caso, pode ser considerada a mesma do aeromodelo, consideramos duas velocidades principais para o projeto: a de decolagem e a de voo em cruzeiro. Ambas foram determinadas através de voos testes com o aeromodelo acoplado a asa de fábrica. Assim, para a decolagem, tem-se um valor médio de 12 m/s e de 17 m/s para o voo horizontal e nivelado.

Para decolagem, onde V = 12 m/s, o número de Reynolds é ReDecolagem = (12 · 0, 32)/(1, 562 · 105) 245.800. Para o voo horizontal e nivelado de cruzeiro, Recruise = (17 · 0, 32)/(1, 562 · 105) 348.200.

Assim, temos uma faixa de número de Reynolds para a análise numérica. Determinou-se que a simulação fosse feita para ângulos de ataques variando entre -3o e 25o e para valores de Reynolds iguais a 245.000 e 349.000.

Usando um nível eletrônico, mediu-se o ângulo no aeromodelo e percebeu-se que sua fuselagem estava inclinada 8o em relação ao solo, e a asa original a 13o . Essa configuração será avaliada durante a corrida para decolagem. Um avião de trem de pouso convencional precisa ganhar velocidade durante a corrida de decolagem para que a força de sustentação levante a parte de trás da aeronave, atingindo assim um ângulo de 0o em relação ao solo, para então, poder fazer o movimento de levantar o nariz do avião (HOMA, 2010). Uma razão de extrema importância é a Cd/Cl, que é a razão entre o coeficiente de sustentação pelo coeficiente de arrasto. Essa razão, quando máxima, indica o valor onde tem-se a maior eficiência em voo, que não necessariamente se assemelha ao valor de máximo Cl. Como buscamos a maior eficiência energética em voo, queremos uma configuração de voo que tenha o menor arrasto possível, solicitando menos potência do motor.

Outro parâmetro importante a ser analisado é a máxima autonomia de voo, que é definida fisicamente, pela razão Cl1,5/Cd em seu ponto de máximo. Esse valor é obtido quando a componente vertical de um voo planado é mínima e, através de manipulação do software XFRL5, é possível obter uma curva da razão Cl1,5/Cd pelo ângulo de ataque.

Em posse dos valores dos ângulos de ataque para a decolagem e do ângulo de ataque para razão Cd/Cl máxima, faremos duas simulações, agora não apenas com o perfil, mas com a asa completa (Ver seção 4.2). Nessa análise serão levados em consideração três parâmetros comuns no estudo de escoamento em uma asa finita: o arrasto induzido, o efeito de downwash e a sustentação.

Referências

- PEREIRA, G. H. S. Análise de parâmetros aerodinâmicos e configuração de sistemas de comunicação e captação de imagem para um VANT com painéis solares nas asas. Faculdade de Tecnologia, Universidade de Brasília, 2019.

1.2.5 Configuração do Piloto Automatico

Nota: Este topico constém trechos replicados de artigos desenvolvidos no Laboratório de Robótica Aérea - UnB. Os artigos são referenciados no fim deste tópico.

A preparação do Pixhawk para voo consiste principalmente em instalar o firmware no dispositivo, conectar e calibrar os sensores e montar na estrutura da aeronave.

Esta seção contém os principais tópicos de configuração e montagem:

QGroundControl

O QGroundControl é uma das principais Estações de Controle em Solo (ECS) disponíveis atualmente a trabalhar com pilotos automáticos compatíveis com MAVLink, incluindo o PX4 e ArduPilot.

Para aplicar neste projeto, escolheu-se o QGroundControl como ECS por fornece uso fácil e direto para iniciantes, além de oferecer, a usuários experientes, suporte a recursos avançados no controle completo de voo e configurações do veículo com PX4.

Além disso, QGroundControl é uma das ECSs mais estáveis em relação as outras, possui uma interface simples e eficiente e está disponível em diversos sistemas operacionais, como Windows, Mac OS X, Linux, Android e o iOS.

Requisitos do Sistema

O QGroundControl pode ser executado normalmente na maioria dos computadores modernos. Um computador com um i5 e pelo menos 8 GB de RAM terá bom desempenho em todos os aplicativos do programa. Para uma melhor experiência, é aconselhável ter o sistema operacional em sua última versão estável.

Instalação

- **Windows**
  - Instalação do QGroundControl nas versões de 64 bits do Windows Vista ou posterior:
    1. Efetue o download do instalador do QGroundControl.exe
    2. Clique duas vezes no executável para inicializar o instalador.
- **Mac OS X**
  - Instalação do QGroundControl no Mac OS 10.10 ou posterior:
    1. Efetue o download do QGroundControl.dmg
    2. Clique duas vezes no arquivo .dmg para instalar
    3. Mova o aplicativo $ QGroundControl para a pasta *Application*
- **Ubuntu**

O Ubuntu possui um gerenciador de modem serial (*serial modem manager*) que interfere nas aplicações envolvendo robótica que utilizam uma porta serial (ou serial USB). Antes da instalação do *QGroundControl* é necessário remover tal gerenciador de modens e conceder ao seu usuário as permissões para acessar a porta serial. Também é preciso instalar o GStreamer para possibilitar o streaming de video.

- Antes de instalar o QGroundControl pela primeira vez:

1. Digite no prompt de comando:

```
sudo usermod -a -G dialout $USER
sudo apt-get remove modemmanager -y
sudo apt install gstreamer1.0-plugins-bad gstreamer1.0-libav -y
```

2. Execute logout e login novamente para ativar a alteração nas permissões de usuario.

- Instalação do QGroundControl no Ubuntu Linux 16.04 LTS ou posterior:

1. Efetue o download do QGroundControl.AppImage
2. Instale e execute os comandos do terminal:

```
cd Downloads
chmod +x ./QGroundControl.AppImage
./QGroundControl.AppImage
```

Dica: A última linha de comando não é necessário se o usuário for ao gerenciador de arquivos, procurar o arquivo QGroundControl baixado e clicar duas vezes nele.

- **Android**
  - O QGroundControl está disponível no Google Play Store em QGroundControl - play.google.com .
- **IOS**
  - O QGroundControl está disponível na App Store.

Firmware

A instalação do *firmware* no hardware do controlador de voo pode ser efetuada de duas formas, pelo uso de um programa de Estação de Controle Terrestre (ECT) ou diretamente pelo uso de ferramentas de desenvolvedor sem o uso de um programa auxiliar[1].

Dica: Antes de iniciar esta seção, recomenda-se o download e instalação do QGroundControl em seu computador.

Nota: A documentação oficial do QGroundControl está disponível em QGroundControl.

Instalação

Recomenda-se a instalação da versão mais recente do PX4, a fim de obter as correções de bug e as melhores e mais recentes funções.

Nota: Antes de instalar o firmware, todas as conexões USB do veículo devem ser desconectadas e o veículo não deve ser alimentado por uma bateria.

---

1. Selecione o icone da engrenagem (**Vehicle Setup**) na barra de ferramentas superior e, em seguida, selecione **Firmware** na barra lateral.
2. Conecte o Pixhawk diretamente ao seu computador via USB (não conecte através de um hub USB).
3. Selecione a opção **PX4 Flight Stack X.x.x Release** para instalar a versão mais recente do PX4 em seu controlador de voo (detectado automaticamente).

[1] Eduardo Moura Cirilo Rocha. 2017. Desenvolvimento de um sistema com veículos aéreos não-tripulados autônomos, Universidade de Brasília, Brasil.

4. Clique em **OK** para iniciar a instalação.

O firmware seguirá com várias etapas de atualização (download do novo firmware, exclusão da versão antiga, etc.). O progresso geral é exibido em uma barra de progresso.

Assim que a instalação acabar e o firmware estiver instalado, o controlador será reiniciado e reconectado.

Mais Informações

- PX4 user guide > Firmware.
- QGroundControl user guide > Firmware.
- PX4 Setup Video (Youtube)

Estrutura da aeronave

Após a instalação do firmware, é necessário configurar os parâmetros do firmware para a estrutura específica do seu veículo.

Definindo a estrutura

1. Com o QGroungControl aberto e o controlador conectado ao computador, selecione o ícone com 3 engrenagens (**Vehicle Setup**) na barra de ferramentas superior e, em seguida, selecione **AirFrame** na barra lateral.
2. Selecione o grupo de veículos que corresponde à estrutura da aeronave e, em seguida, utilize o menu suspenso dentro do grupo para escolher o tipo de estrutura que melhor corresponde ao seu veículo.
3. Clique em **Aplly and Restart**. Em seguida, vá em **Apply** no *prompt* para salvar as configurações e reiniciar o veículo.

Mais informações

- PX4 user guide > Airframe.
- QGroundControl user guide > Airframe.

Conexões

A imagem abaixo apresenta as conexões dos sensores e demais itens inclusos no Pixhawk. Cada parte será analisada com mais detalhes nas seções a seguir.

Campainha e interruptor de segurança

A campainha fornece sinais sonoros que indicam a situação do VANT. Enquanto o interruptor atua na segurança da aeronave, bloqueando e desbloqueando os motores.

Nota: O interruptor de segurança é ativado por padrão e quando ativado, não permite o voo, bloqueando os motores. Para desativar o modo de segurança, pressione e segure o interruptor por 1 segundo. Você pode ativar o modo de segurança novamente pressionando o interruptor.

Para conectar a campainha e o interruptor de segurança (itens obrigatórios), basta liga-lós ao Pixhawk como mostrado abaixo.

Divisor I2C

O *slitter* I2C expande a quantidade de portas I2C permitindo a conexão de até quatro periféricos ao Piwhawk. Utilize um cabo de 4 fios para conectar o *slitter* I2C e para alimentar uma bússola externa, um display LED, um sensor de velocidade do ar digital e/ou qualquer outro periférico compatível ao veículo.

Sensor de velocidade do ar

Em edição...

GPS + Compass

O GPS, outro dispositivo obrigatório, deve ser conectado à porta GPS (6 pinos) usando o cabo de 6 fios fornecidos no kit. A conexão da bússola é opcional, porém recomendamos fortemente sua utilização. Para conecta-lá, ligue um cabo de 4 fios a uma porta I2C do *slitter* I2C, como mostrado abaixo.

Nota: O GPS/bússola deve ser montado no chassi da aeronave o mais longe possível de outros componentes eletrônicos, com a seta indicadora voltada para a frente e o mais alinhada possível com o Pixhawk.

Rádio controle

O sistema de rádio controle (RC) é necessário caso deseje controlar manualmente seu veículo, dado que o Pixhawk não requer um sistema de rádio para modos de voo autônomo.

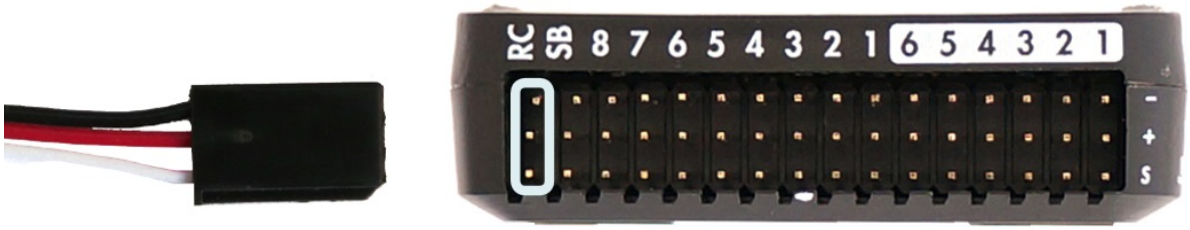
Para conectar o sistema de rádio controle, será necessário selecionar um transmissor/receptor compatível e depois vinculá-lo para que eles se comuniquem.

Dica: Leia as instruções que acompanham seu transmissor/receptor.

As instruções a seguir mostram como conectar os diferentes tipos de receptores ao Pixhawk:

- Os receptores Spektrum e DSM se conectam à entrada SPKT/DSM .

- Os receptores PPM-SUM e S.BUS conectam-se aos pinos de terra, potência e sinal RC, conforme mostrado.

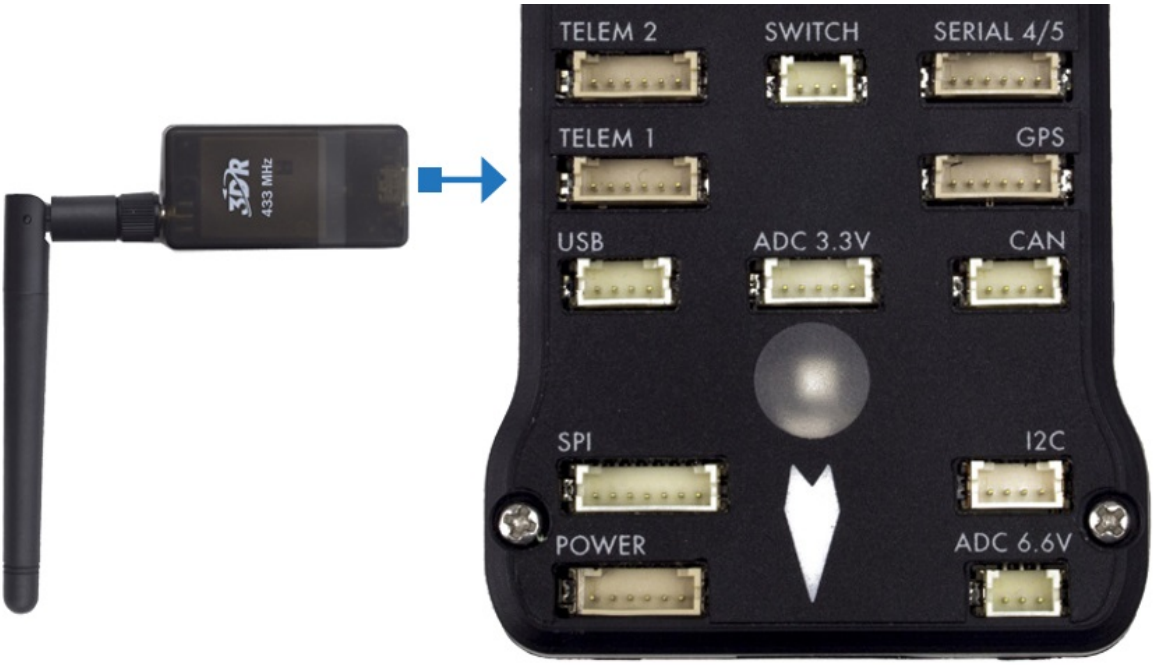


- Os receptores PPM e PWM que possuem um fio individual para cada canal devem se conectar à porta RC por meio de um codificador PPM (os receptores PPM-Sum usam um único fio de sinal para todos os canais).

Para obter mais informações sobre a seleção de um sistema de rádio, a compatibilidade do receptor e a ligação do seu par transmissor e receptor, consulte: Transmissores e receptores de controle remoto.

Telemetria

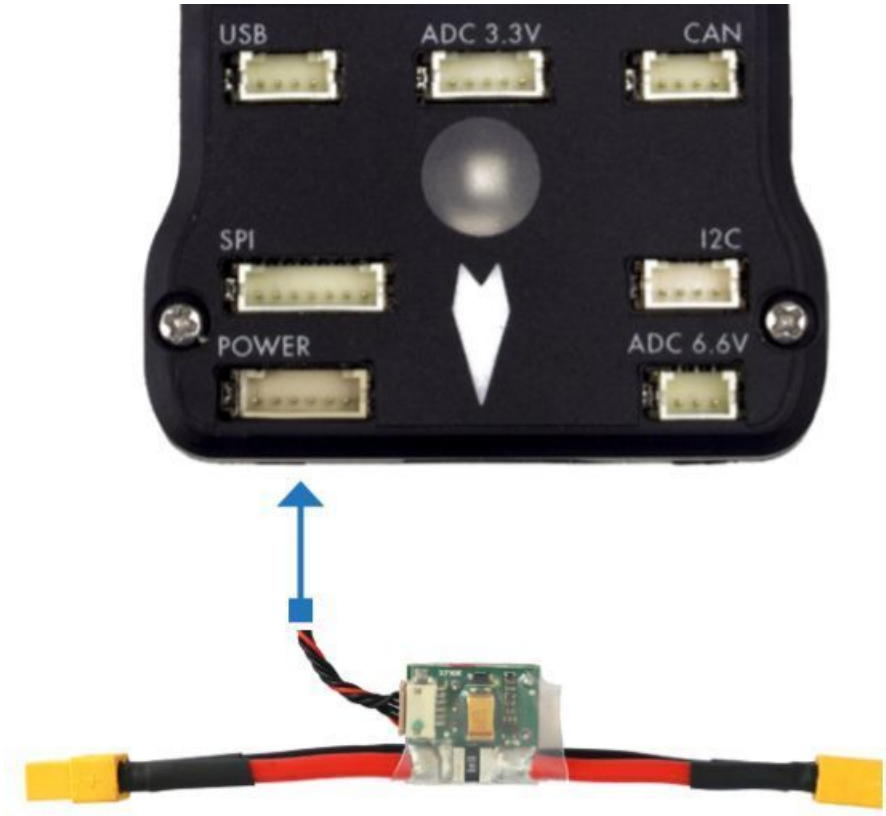
Os modens de telemetria podem ser usados para comunicar e controlar um veículo em voo a partir de uma estação terrestre (por exemplo, você pode direcionar o VANT para uma posição específica ou carregar uma nova missão). Um modem deve ser conectado ao seu veículo, como mostrado abaixo. O outro modem deverá ser conectado ao computador da estação terrestre ou dispositivo móvel (geralmente por uma porta USB).



Módulo de energia

O módulo de energia (*Power module* - PM) fornece energia ao controlador de voo da bateria e também envia informações sobre a corrente analógica e a tensão fornecida pelo módulo (incluindo a energia do controlador de voo e dos motores, etc.).

A saída do modulo de energia (PM) deve ser conectada à porta **POWER** do Pixhawk usando um cabo de 6 fios, como apresentado na imagem. A entrada do modulo deverá ser conectada a uma bateria de LiPo, enquanto a saída principal será responsável por fornecer energia aos ESCs e motores da aeronave (possivelmente através de uma placa de distribuição de energia, a depender da aeronave).



Sensor de distancia

O Pixhawk suporta vários sensores de distância diferentes, incluindo os Lidars (que usam lasers ou raios infravermelhos para medições de distância) e Sonars (que utilizam som ultrassônico), e também incluem os buscadores de alcance LED Maxbotix Sonar e Pulsed Light. Dessa forma, a instalação varia de dispositivo para dispositivo. Mais informações a respeito da configuração dos sensores pode ser visualizada em Rangefinders.

Fig. 39: Exemplo de alguns sensores de distância compatíveis

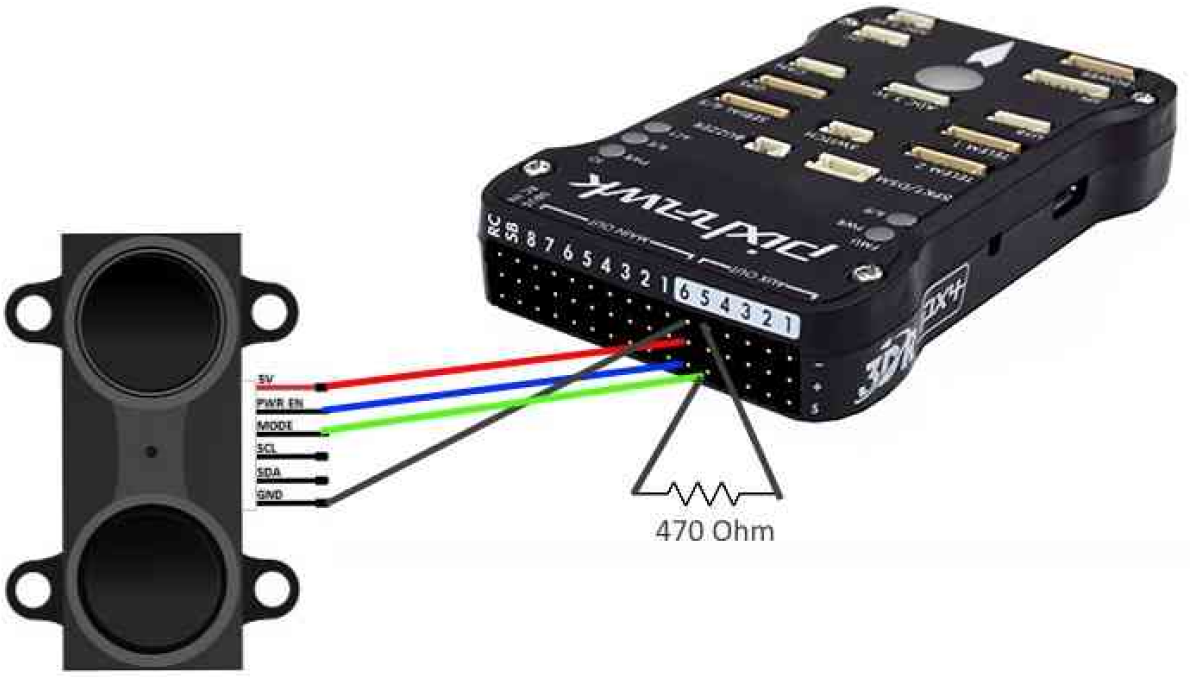
Para implementar o projeto, escolheu-se o sensor Lidar para habilitar a função de pouso automático devido sua maior precisão em relação aos demais. O sensor lidar pode ser conectado ao Pixhawk de duas formas, através do protocolo I2C na porta I2C (ou I2C *slitter*) ou por *pulse-width-modulation* (PWM) na trilha PWM.

De acordo com a documentação do Pixhawk, o lidar utilizado apresenta problemas de interferência com outros dispositivos quando conectado na porta I2C. Assim, escolheu-se a conexão por PWM. Um diagrama de conexão pode ser vista na tabela abaixo e o esquema de montagem pode ser visto na figura a seguir, onde o valor do resistor pode variar entre 200Ω e 1kΩ[1].

[1] Eduardo Moura Cirilo Rocha. 2017. Desenvolvimento de um sistema com veículos aéreos não-tripulados autônomos, Universidade de Brasília, Brasil

Table 7: Diagrama de conexão entre o Lidar e o Pixhawk

Sinal LIDAR-Lite	Sinal Pixhawk
J1	CH6 Out - V+
J2	CH6 Out - Signal (sinal interno 55)
J3	CH5 Out - Signal (sinal interno 54)
J4	
J5	
J6	Ch6 Out - Ground



Mais detalhes sobre a conexão podem ser encontrados em LIDAR-Lite Rangefinder.

Mais informações e referências

- Pixhawk Wiring Quick Start - PX4 User Guide
- Basic Assembly - PX4 User Guide
- Pixhawk Series - PX4 User Guide
- Peripheral Hardware - Ardupilot Docs

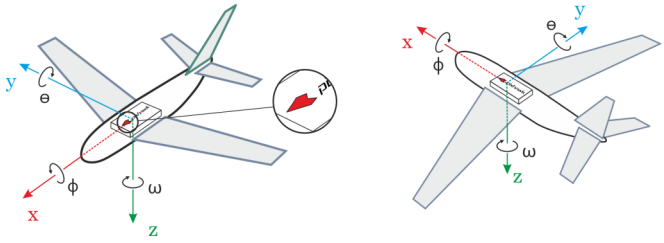
Montando o Pixhawk

Orientação do Piloto Automático

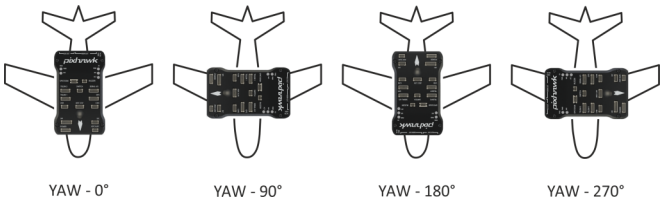
Por padrão, o controlador de voo e a bússola externa devem ser colocados na estrutura da aeronave orientados de modo que a seta aponte para a frente do veículo. Se a placa ou a bússola externa estiverem em qualquer outra orientação, será necessário corrigir a orientação no firmware.

Cálculo da Orientação

As compensações dos ângulos de rotação YAW, PITCH e / ou ROLL são calculados em relação à orientação vertical apontando para a frente (rotação no sentido horário em torno dos eixos Z, Y e X, respectivamente). Esse diagrama é chamado de *body frame* (diagrama de corpo) e a orientação padrão é dada por `ROTATION_NOME`.



Por exemplo, a imagem abaixo apresenta rotações de aeronaves em torno do eixo z (YAW), correspondendo, respectivamente, a: `ROTATION_NONE`, `ROTATION_YAW_90`, `ROTATION_YAW_180`, `ROTATION_YAW_270`.



Definindo a Orientação

Para definir as orientações no firmware:

Nota: Antes de definir a orientação, o QGroundControl deve ser iniciado, conectado ao veículo e o firmware já deve ter sido instalado na placa controladora de voo.

---

1. Selecione o ícone de **engrenagem** (Configuração do veículo) na barra de ferramentas superior e, em seguida, **Sensors** na barra lateral.
2. Selecione o botão **Set Orientations**.
3. Selecione a orientação do piloto automático, conforme calculado.
4. Selecione a **External Compass Orientation** (Orientação da bússola externa) da mesma maneira (esta opção será exibida apenas se o seu veículo tiver uma bússola externa).
5. Pressione **OK**.

Dica: A documentação completa sobre como ajustar a orientação do piloto automático está disponível em Orientação do piloto automático.

Isolamento de Vibrações

As placas Pixhawk possuem acelerômetros e giroscópios embutidos, sendo sensíveis a vibrações. Elevados níveis de vibração podem causar uma série de problemas, incluindo redução do desempenho de voo, voos mais curtos e maior desgaste do veículo. Em casos extremos, a vibração pode levar a falhas dos sensores, resultando em falhas de estimativa ou até mesmo a interrupção do voo.

Por essa razão, o Pixhawk vem com *espumas de amortecimento de vibrações*.

O Pixhawk deve ser montado na aeronave utilizando as espumas antivibratórias incluídas no kit. Ele deve ser posicionado o mais próximo possível do centro de gravidade do veículo.

Dica: Para determinar se os níveis de vibração estão muito altos e utilizar algumas técnicas para melhorar as características de vibração, recomenda-se o tópico PX4 user guide > Vibration Isolation.

Mais Informações

- Advanced Orientation Tuning.
- PX4 user guide > Sensor Orientation.
- QGroundControl user guide > Sensors.
- PX4 user guide > Vibration Isolation.

Calibração do Piloto Automático

Esta seção contém os principais tópicos de calibração do piloto automático:

Calibração da Bússola

Todos os magnetômetros internos e externos conectados ao Pixhawk serão configurados no processo de calibração da bússola

Etapas da Calibração

1. Abra o aplicativo QGroundControl e conecte o veículo pelo fio ao usb do computador.
2. Selecione o ícone **Engrenagem** (Configuração do veículo) na barra de ferramentas superior e, em seguida, **Sensores** na barra lateral.
3. Clique no botão do sensor da **Bússola**.
4. Clique **OK** para começar a calibração.
5. Coloque o veículo em umas das posições indicadas em vermelho (não calibrada), mantenha essa posição escolhida por alguns instantes. Uma vez solicitado (a imagem de orientação fica amarela), gire o veículo em torno do eixo especificado nas duas direções. Quando a calibração estiver concluída para a orientação atual, a imagem associada na tela ficará verde.
6. Repita o processo de calibração para todas as orientações do veículo.

Mais Informações

- PX4 user guide > Firmware.
- QGroundControl user guide > Sensors.
- PX4 Setup Video (Youtube)

Calibração do Giroscópio

Pelo aplicativo QGroundControl será orientado a posicionar o veículo em uma local com uma superfície plana e mantê-lo imóvel na posiçao determinada.

Etapas da Calibração

1. Clique no botão do sensor do **Giroscópio**.
2. Posicione o veículo sobre a superfície plana e mantenha-o na posição.
3. Clique **OK** para começar a calibração.
4. Quando finalizado, o aplicativo QGroundControl mostrará uma barra verde completa e uma contorno verde ao redor da imagem do veículo.

Dica: Caso ocorra algum movimento no veículo, o aplicativo QGroundControl irá automaticamente reiniciar o processo de calibração do giroscópio.

Mais informações

- QGroundControl user guide > Gyroscope.

Calibração do Acelerometro

O aplicativo QGroundControl mostrará todos os passos para colocar e segurar o veículo em diversas orientações.

Etapas da Calibração

Dica: É necessário ter a orientação do piloto automático definida para prosseguir com as etapas de calibração do acelerômetro. Caso contrário, entre na aba **Mountig the Pixhawk** e execute as etapas na subaba **Setting the Orientation**.

---

1. Abra o aplicativo QGroundControl e conecte o veículo pelo fio ao usb do computador.
2. Selecione o ícone **Engrenagem** (Configuração do veículo) na barra de ferramentas superior e, em seguida, **Sensores** na barra lateral.
3. Clique no botão do sensor do **Acelerometro**.
4. Clique **OK** para começar a calibração.
5. Coloque o veículo nas posições mostradas pela imagem que aparece na tela do aplicativo. Quando a posição é solicitada, a imagem de orientação fica amarela, deixe-a nessa posição (sem mover o veículo) até que a calibração seja concluída. A imagem ficará verde quando a calibração estiver concluída.
6. Repita o processo de calibração para todas as orientações do veículo.

Mais Informações

- PX4 user guide > Accelerometer.
- QGroundControl user guide > Sensors.
- PX4 Setup Video (Youtube)

Calibragem do Tubo de Pito

A calibração da velocidade do ar precisa ler uma linha de base estável com 0 velocidade do ar para determinar um deslocamento. Coloque as mãos sobre o pitot para bloquear qualquer vento (se não for necessário calibrar o sensor em ambientes fechados) e sopre o tubo com a boca (para sinalizar a conclusão da calibração).

Etapas da Calibração

1. Abra o aplicativo QGroundControl e conecte o veículo pelo fio ao usb do computador.
2. Selecione o ícone **Engrenagem** (Configuração do veículo) na barra de ferramentas superior e, em seguida, **Sensores** na barra lateral.
3. Clique no botão do sensor do **AirSpeed**.
4. Proteja o sensor do vento (ou seja, cubra-o com a mão). Tome cuidado para não bloquear nenhum dos orifícios.
5. Clique **OK** para começar a calibração.
6. Sopre na ponta do tubo pitot para indicar que a calibração está concluída.
7. Aguarde dois a três segundos antes de remover a proteção contra o vento, lembrando que a calibração será concluída em alguns segundos.

Dica: Soprar no tubo também é uma verificação básica de que as portas estáticas e dinâmicas estão instaladas corretamente. Se eles forem trocados, o sensor exibirá uma grande pressão diferencial negativa quando você soprar no tubo, e a calibração será interrompida com um erro.

Mais Informações

- QGroundControl user guide > AirSpeed.

Calibração do Radio

A opção Radio Setup, presente no QGroundControl, é usada para configurar o mapeamento dos principais bastões de controle de atitude da unidade de controle remoto (rotation, pitch, yaw, throttle) para os canais e calibrar as configurações mínimas, máximas, de compensação e reversão para todos os outros controles / canais RC.

Conexão Radio-Receptor-Transmissor

Antes de calibrar o sistema de rádio, o receptor e o transmissor devem estar conectados. O processo de conexão de um transmissor e receptor é específico do hardware.

- QGroundControl user guide > Spektrum receiver.
- FrSky receiver (Youtube)

Etapas de Calibração

No processo de calibração, será indicado que os bastões são movidos em um padrão específico que é mostrado no diagrama do transmissor no canto superior direito da tela.

1. Ligue o transmissor RC.
2. Abra o aplicativo QGroundControl e conecte o veículo pelo fio ao usb do computador.

3. Selecione o ícone **Engrenagem** (Configuração do veículo) na barra de ferramentas superior e, em seguida, **Rádio** na barra lateral.
4. Pressione **OK** para iniciar a calibração.
5. Defina o botão de opção do modo de transmissor que corresponde ao seu transmissor (isso garante que o QGroundControl exiba as posições corretas dos bastões para você seguir durante a calibração).
6. Mova os bastões para as posições indicadas no texto (e na imagem do transmissor). Pressione **Next** quando os bastões estiverem na posição. Repita para todas as posições.
7. Quando solicitado, mova todos os outros por toda as possibilidades de posições (você poderá observá-los se movendo no Monitor de canal).
8. Pressione **Avançar** para salvar as configurações.

Mais Informações

- Radio Setup Video (Youtube)

Pouso Automático

Após a conexão do Lidar ao sistema via PWM, alguns parâmetros do piloto automático devem ser alterados para que ele reconheça o sensor. Esses parâmetros podem ser facilmente alterados através do QGroundControl. São eles:

- RNG_FND = 5, indica que a conexão ocorre via PWM.
- RNDFND_MAX_CM = 4000, representa a distância máxima em que o sensor é confiável.
- RNDFND_STOP_PIN = 55, indica o pino conectado ao sinal de ativação do Lidar. Permite que o dispositivo reinicie o sensor caso ele para de fornecer dados.
- Os parâmetros RNDFND_SCALING e RNDFND_OFFSET devem ser ajustados de forma a se calibrar o sensor (costumam ser aproximadamente 0 e 1, respectivamente).

O sensor pode ser testado pelo QGroundControl, onde as leituras podem ser observadas no campo **Sonar Range**. Após a configuração do sensor, o piloto automático será capaz de pousar a aeronave de forma muito mais rápida e precisa. O pouso ocorre pelo envio do comando **Land** ao controlador, mas para que ele ocorra corretamente deve-se definir a posição da pista de pouso e deve-se ajustar os parâmetros de pouso, como, por exemplo, a velocidade com que o avião deve pousar.

A documentação detalhada sobre pousos automáticos pode ser encontrada em Automatic Landing.

1.2.6 Sistemas de comunicação

Nota: Este topico constém trechos replicados de artigos desenvolvidos no Laboratório de Robótica Aérea - UnB. Os artigos são referenciados no fim deste tópico.

Para monitoramento e controle das atividades em vôo se faz necessário uso de uma Estação de Controle Terrestre (ECT) por meio do protocolo de comunicação MAVLink, Micro Air Vehicle Communication Protocol, especialmente para uso em veículos aéreos de pequeno porte. Uma ECT é uma aplicação de software para uso em computador capaz de se comunicar com um VANT por meio da telemetria, mostrando os dados obtidos em tempo real pelo piloto automático. Por meio desse monitoramente, é possível a tomada de decisão para executar ações no VANT por meio da telemetria.

O protocolo MAVLink é capaz de enviar estruturas de dados, no formato de sequência de bytes, em canais seriais para a estação de controle em terra. Esses dados são recebidos pelo piloto automático ou pela estação de controle por meio de conexão USB ou por comunicação por telemetria. Após o recebimentos desses dados, o software da ECT é responsável pela decodificação dos mesmos permitindo a interpretação do operador.

As principais ECTs disponíveis são o MissionPlanner, APM Planner e QGroundControl. Neste trabalho a ECT utilizada será a MissionPlanner devido a vasta biblioteca de material introdutório bem como suporte de outros estudantes. Também no sistema de Navegação, foi implementado modo de vôo com FPV, first person view, com observação em primeira pessoa na estação de controle. A transmissão de vídeo para a estação é feita independente do piloto automático, sua alimentação é feita diretamente na bateria e é transmitida através de receptores na faixa de 5,8 GHz.

Referências

- GOMES, D. H. Configuração e estudo do sistema de gerenciamento de energia, navegação e controle como parâmetros ótimos na montagem e desempenho de um VANT com painéis solares. Faculdade de Tecnologia, Universidade de Brasília, 2019.
- ROCHA, Eduardo M. C, (2017). Desenvolvimento de um sistema com veículos aéreos nãotripulados autônomos, Universidade de Brasília, Trabalho de Graduação 1.

1.2.7 Sistema de FPV

Nota: Este topico constém trechos replicados de artigos desenvolvidos no Laboratório de Robótica Aérea - UnB. Os artigos são referenciados no fim deste tópico.

Para o funcionamento do sistema de comunicação é preciso ter uma base emissora e uma base receptora. A base receptora escolhida foi um computador com o software Mission Planner instalado, que será responsável pelo recebimento e pela exibição dos dados de voo, como ângulo de atitude, nível da bateria, potência do motor entre outros.

Para a base emissora, existe duas possibilidades de conexão: por USB ou por telemetria. A conexão por USB é simples e é feita conectando o computador da base direto na Pixhawk, responsável pelo piloto automático. Essa opção foi utilizando quando se precisava fazer testes em solo, mas inviável quando em voo. Em voo, utilizou-se a telemetria para a transmissão desses dados, tendo como forma de transmissão uma antena localizada na região do nariz do aeromodelo ligada a Pixhawk.

O sistema de aquisição de imagem é do tipo FPV. Esse tipo de sistema vem com um transmissor, que deve ser colocado dento do aeromodelo, e um receptor de deverá estar conectado a um monitor em solo. Com isso, é possível obter imagens em tempo real do voo, sendo necessário apenas que o transmissor esteja ligado a bateria do aeromodelo. A marca do FPV utilizado nesse projeto é a Fat Shark, modelo Pilot Cam 700CCD, pois ela apresenta uma qualidade razoável de imagem e por isso, atende os requisitos de missão.

Para esse sistema, foram realizados apenas testes em laboratório, ou seja, o sistema não foi inserido no aeromodelo em nenhum momento durante os voos de teste no parque de aeromodelismo.

Referências

- PEREIRA, G. H. S. Análise de parâmetros aerodinâmicos e configuração de sistemas de comunicação e captação de imagem para um VANT com painéis solares nas asas. Faculdade de Tecnologia, Universidade de Brasília, 2019.

1.2.8 Referências

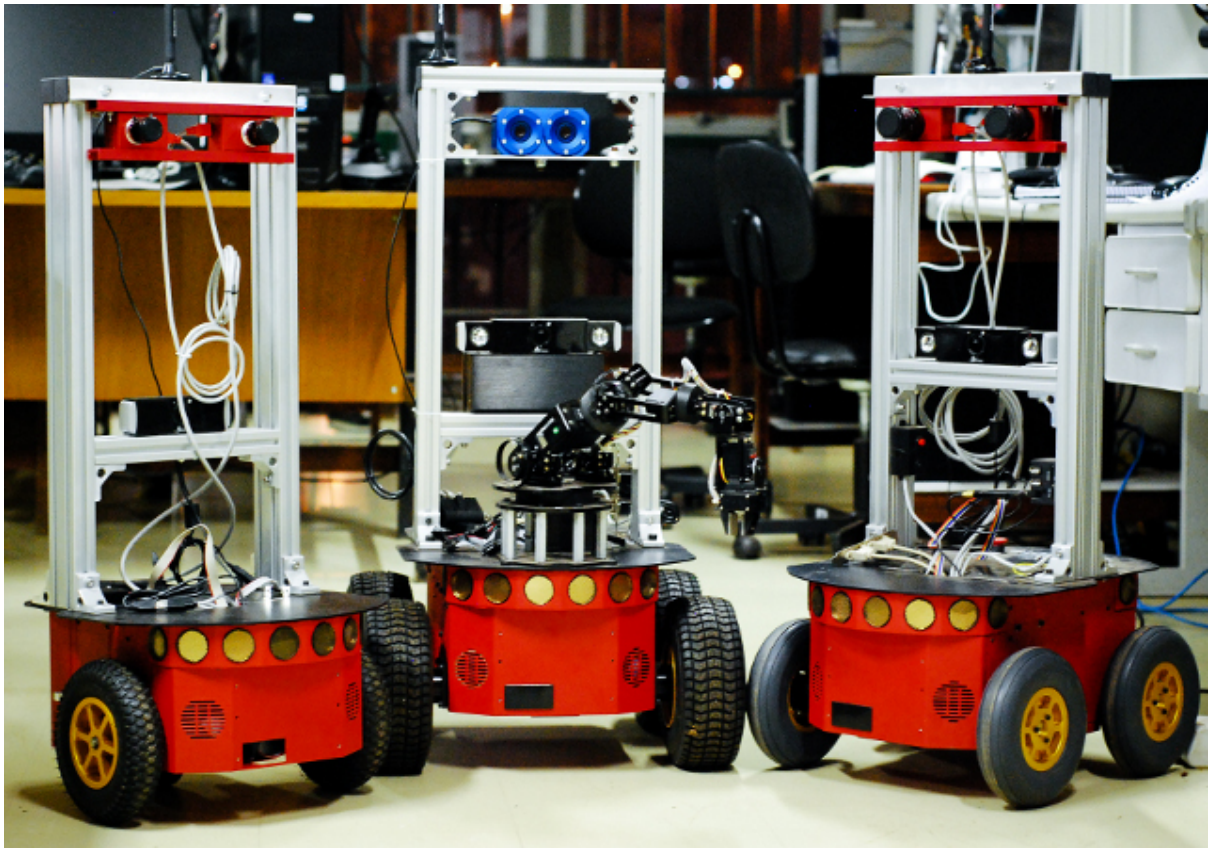
- PEREIRA, G. H. S. Análise de parâmetros aerodinâmicos e configuração de sistemas de comunicação e captação de imagem para um VANT com painéis solares nas asas. Faculdade de Tecnologia, Universidade de Brasília, 2019.
- GOMES, D. H. Configuração e estudo do sistema de gerenciamento de energia, navegação e controle como parâmetros ótimos na montagem e desempenho de um VANT com painéis solares. Faculdade de Tecnologia, Universidade de Brasília, 2019.

- PX4 Autopilot User Guide - docs.px4.io
- QGroundControl User Guide - qgroundcontrol.com
- Ardupilot Docs - ardupilot.org
- RT-MaG Project - gipsa-lab.fr
- Yocto Project - yoctoproject.org
- Getting Started - Gumstix COM - gumstix.com
- Gumstix, Inc - GitHub - github.com

CAPÍTULO 2

Mobile Robots

2.1 Pioneer



2.1.1 Kinematics

Kinematics, in classical mechanics, study the description of the motion of points, bodies (objects), and groups of bodies without considering the forces that cause them to move. In mobile robotics, kinematics helps us to understand and quantify the constraints about the robot design, which

implies restrictions in its movement. Also, we can draw the paths and trajectories that the robot can do. Here, the text focus on wheeled robots, which can only move in 2D space.

Fig. 2: Pioneer P3-AT

The P3-DX is a two-wheel-drive and P3-AT is a four-wheel-drive. The Omron Adept MobileRobots considers in its manual the P3-DX as a differential drive robot and the P3-AT as a skid/slip drive robot. For our sake, both are modeled as differential drive vehicles because the exact center of rotation in a skid/slip drive is unpredictable[1].

In order to represent the motion of a mobile robot, we must define the reference frames and determine their position. There are two essential frames to a robot if we consider the robot as a rigid body. They are the global reference frame, that is world fixed, and the local reference frame, which is robot fixed.

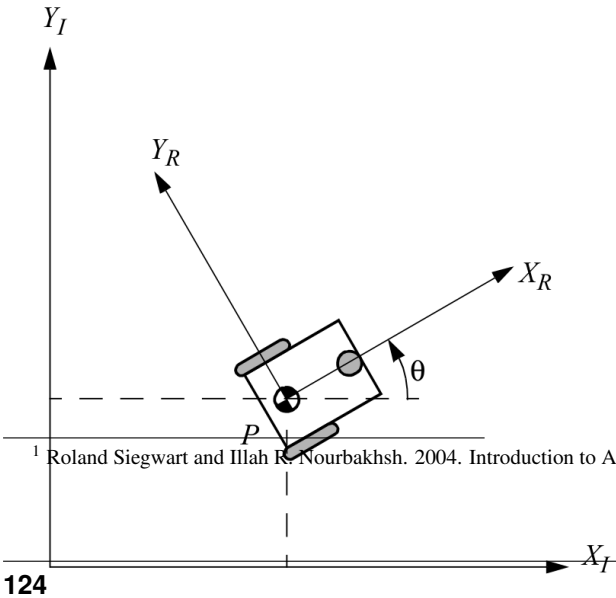


Fig. 3: The global reference frame and the robot local reference frame. Figure from[1].

The figure shows a robot and its reference frames. Where the X_I and Y_I defines the global reference frame, also known as the inertial frame, and X_R and Y_R defines the local reference frame or the robot frame. The coordinates x and y represent the robot's position in the global reference frame, point P, whereas θ is the angular difference among the global and the local reference frames. Thus, we represent the robot's pose as the vector with these three components.

$$\xi_I = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

[1] Roland Siegwart and Illah R. Nourbakhsh. 2004. Introduction to Autonomous Mobile Robots. Bradford Company, USA.

Nota: We could still assume the robot as a rigid body free in the space, with 6 degrees of freedom, $[x, y, z, \phi, \psi, \theta]^T$. However, as the robot is moving subject to gravity, which keeps it confined to a euclidian plane, x and y describe the robot's position, and θ describes the orientation in the plane. The 2D space in which the robot lies is called the *C-space*[5], firstly formalized in[6], also called the configuration space. A configuration is a complete specification of the position of every point in the system. The space of all configurations is the *C-space*.

We can now utilize this definition to describe elements represented in the local frame in the global frame and vice-versa. For example, we can map the motion calculated in the global frame to motion in the robot's local frame. Or, we can map obstacles sensed by the robot in terms of the global reference frame.

$$R(\theta) = \begin{bmatrix} cos\theta \\ sin\theta \\ 0 \\ -sin\theta \\ cos\theta \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

The *orthogonal rotation matrix* does the map between these frames as a function of the robot's current pose.

$$\dot{\xi_R} = R(\theta)\dot{\xi_I}$$

Nota: It does not make sense to talk about the robot's pose in the local frame. Because, the point P, the origin of the coordinates, moves around with the robot. So, the robot's pose in the local frame is, always, $\xi_R = [0, 0, 0]^T$. That is why the relationship is between the derivative of the pose in the frames.

Wheels and its constraints

«Wheeled Mobile Robots (WMR) constitute a class of mechanical systems characterized by kinematics constraints that are not integrable and cannot, therefore, be eliminated from the model equations»[2]. If we want to study and describe a robot motion, we also must specify which are the hypotheses and constraints of the wheels. There are three essential hypotheses about the kinematics model of the wheeled robot during the motion; they are:

- Each wheel remain perpendicular about its plane;
- There is only one contact point between plane and wheel;
- There is only rolling without slipping;

And two constraints:

- About rolling: the motion component along the wheel plane is equal to the rotation velocity of the wheel;

[5] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki and S. Thrun, «Principles of Robot Motion: Theory, Algorithms, and Implementations,» MIT Press, Boston, 2005.

[6] Lozano-Perez, «Spatial Planning: A Configuration Space Approach,» in IEEE Transactions on Computers, vol. C-32, no. 2, pp. 108-120, Feb. 1983.

[2] G. Campion, G. Bastin and B. Dandrea-Novel, «Structural properties and classification of kinematic and dynamic models of wheeled mobile robots,» in IEEE Transactions on Robotics and Automation, vol. 12, no. 1, pp. 47-62, Feb. 1996.

- About slipping: the motion component along the orthogonal direction is equal to zero;

Some authors may call these constraints as «the pure rolling and rotation condition».

Differential Drive Model

Now we delve into modeling of a differential drive kinematic. Dudek et al. say that the differential drive consists of two wheels mounted in the same axis with separated motors[3]. Each wheel contributes to the robot motion, so to fully describe the robot motion, we must compute each contribution.

The image shows the robot, the wheel velocities, and the local frame. $\dot{\phi}_1$ and $\dot{\phi}_2$ is the spin speed of the left wheel and right wheel. r is the wheel radius, while the distance between the two wheels is l. While v_1 is the left wheel velocity along the ground and v_2 the right wheel velocity. As the wheels contribute independently to the robot motion, we can analyze each contribution separately.

$$v_i = \frac{\dot{\phi}_i r}{2}$$
$$\omega_i = \frac{\dot{\phi}_i r}{2l}$$
$$\text{where } i = \{1, 2\}$$

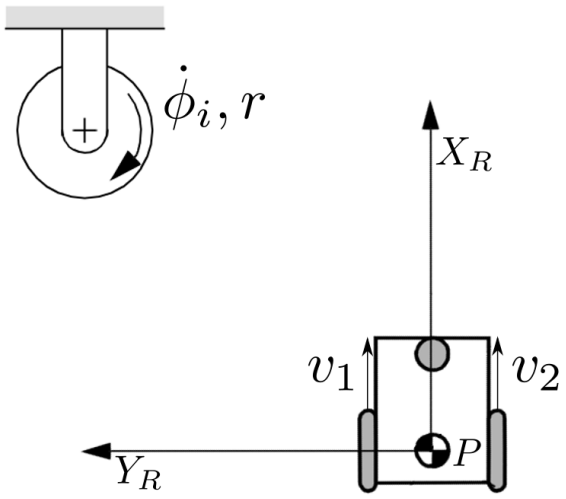


Fig. 4: Wheel velocities and the robot frame.

Point P is halfway between the two wheels, so each wheel contributes with half of the linear speed of the robot in the direction of X_R. Each wheel also adds a new component to the angular speed of the robot. v_1 moves the robot clockwise around point P while v_2 moves it counter-clockwise. That is why they differ in their sign. And, using the equation which relates the angular speed of disk with its linear speed, we have the above equations.

Using the superposition theorem, we have the equations for the linear velocity in the direction of X_R and the angular velocity in the direction of Z_R:

$$v = v_1 + v_2$$
$$\omega = -\omega_1 + \omega_2$$

[3] Gregory Dudek and Michael Jenkin. 2010. Computational Principles of Mobile Robotics (2nd. ed.). Cambridge University Press, USA.

In the local frame, we have the following kinematic equation:

$$\dot{\xi_R} = \begin{bmatrix} \frac{r}{2} \\ \frac{r}{2} \\ 0 \\ 0 \\ -\frac{r}{2l} \\ \frac{r}{2l} \end{bmatrix} \begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \end{bmatrix}$$

Nota: In the robot frame, there is no velocity in the direction of Y_R. Because we assumed the pure rolling and rotation condition. And yet he can reach any point in the global frame.

Forward Kinematics

The forward kinematics problem tries to solve the problem when we have the control inputs, and we must know where the robot goes in the global frame. As we have seen, to solve this question, we should know five parameters of the robot — two parameters about the robot geometry, l and r, the current robot orientation, θ, and, at least, the two inputs, $\dot{\phi}_1$ and $\dot{\phi}_2$.

$$\dot{\xi_I} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f(l, r, \theta, \dot{\phi_1}, \dot{\phi_2})$$

f is the function that solves the forward kinematics problem. To map between the parameter vector, $\{l, r, \theta, \phi_1, \phi_2\}$, and the state of the robot in the inertial frame. We should use the matrix, which links the spin speed and the derivative of the robot state in the local frame. Then, we can transform the robot velocities in the local frame to the global frame utilizing the inverse of the rotation matrix.

$$R(\theta)^{-1} = \begin{bmatrix} cos\theta \\ -sin\theta \\ 0 \\ sin\theta \\ cos\theta \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

$$\dot{\xi_I} = R(\theta)^{-1}\dot{\xi_R},$$

$$\dot{\xi_R} = \begin{bmatrix} \frac{r}{2} \\ \frac{r}{2} \\ 0 \\ 0 \\ -\frac{r}{2l} \\ \frac{r}{2l} \end{bmatrix} \begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \end{bmatrix}$$

$$\dot{\xi_I} = R(\theta)^{-1} \begin{bmatrix} \frac{r}{2} \\ \frac{r}{2} \\ 0 \\ 0 \\ -\frac{r}{2l} \\ \frac{r}{2l} \end{bmatrix} \begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \end{bmatrix}$$

Then,

$$f(l, r, \theta, \dot{\phi}_1, \dot{\phi}_2) =$$

$$\begin{bmatrix} cos\theta \\ -sin\theta \\ 0 \\ sin\theta \\ cos\theta \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} \frac{r}{2} \\ \frac{r}{2} \\ 0 \\ 0 \\ -\frac{r}{2l} \\ \frac{r}{2l} \end{bmatrix} \begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \end{bmatrix}$$

$$f(l, r, \theta, \dot{\phi}_1, \dot{\phi}_2) =$$

$$\begin{bmatrix} \frac{rcos\theta}{2} \\ \frac{rcos\theta}{2} \\ \frac{rsin\theta}{2} \\ \frac{rsin\theta}{2} \\ -\frac{r}{2l} \\ \frac{r}{2l} \end{bmatrix} \begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \end{bmatrix}$$

Or

$$\dot{\xi}_I =$$

$$\begin{bmatrix} \frac{rcos\theta}{2} \\ \frac{rcos\theta}{2} \\ \frac{rsin\theta}{2} \\ \frac{rsin\theta}{2} \\ -\frac{r}{2l} \\ \frac{r}{2l} \end{bmatrix} \begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \end{bmatrix}$$

Nota: The matrix which maps spin speed to the robot velocities is commonly known as **Jacobian Matrix**. «The Jacobian maps configuration velocities to workspace velocities»[5].

Well, we know the relationship between spin speeds and robot velocities, but what about the robot pose in the global frame?

$$\xi_I = \int_0^t \begin{bmatrix} \frac{rcos\theta}{2} \\ \frac{rcos\theta}{2} \\ \frac{rsin\theta}{2} \\ \frac{rsin\theta}{2} \\ -\frac{r}{2l} \\ \frac{r}{2l} \end{bmatrix} \begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \end{bmatrix} dt$$

Or

$$\begin{cases} x(t) = \frac{r}{2} \int_0^t (\dot{\phi}_1(t) + \dot{\phi}_2(t)) cos(\theta(t)) dt \\ y(t) = \frac{r}{2} \int_0^t (\dot{\phi}_1(t) + \dot{\phi}_2(t)) sin(\theta(t)) dt \\ \theta(t) = \frac{r}{2l} \int_0^t (\dot{\phi}_2(t) - \dot{\phi}_1(t)) dt \end{cases}$$

Inverse Kinematics

The inverse kinematics problem is the opposite of the forward problem. The problem aims to solve the following question: «Given the desired pose, which are the controls needed to reach the desired pose?». We already know the relationship between the velocity and

$$\begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \end{bmatrix} = g(\dot{\xi}_I)$$

The function g is the mathematical inverse of the function f.

$$g = f^{-1} = \begin{bmatrix} \frac{rcos\theta}{2} & \frac{rcos\theta}{2} \\ \frac{rsin\theta}{2} & \frac{rsin\theta}{2} \\ -\frac{r}{2l} & \frac{r}{2l} \end{bmatrix}^{-1}$$

As we can see, the matrix which represents the function f is not invertible. The forward kinematics is an easy problem because we have one and only one solution. Nevertheless, the inverse kinematics is often not analytically solvable; commonly, we have more than one solution or none. However, we can try to solve the problem, limiting the possibles solutions like $\dot{\phi}_1 = \dot{\phi}_2$ or $\dot{\phi}_1 = -\dot{\phi}_2$.

Straight Line

If we limit the solution to $\dot{\phi}_1 = \dot{\phi}_2 = \dot{\phi}$, with $\dot{\phi} > 0$, the robot should move along a straight line. Then, the robot motion simplifies to:

$$\xi_I = \begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x + vcos(\theta)\delta t \\ y + vsin(\theta)\delta t \\ \theta \end{bmatrix}$$

Rotaion in place

Similarly, if we limit the solution to $-\dot{\phi}_1 = \dot{\phi}_2$, with $\dot{\phi}_2 > 0$, the robot should rotate in the place around the point P.

$$\xi_I = \begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta + \frac{2v}{l}\delta t \end{bmatrix}$$

Motion Composition

If we would like to drive the robot from any pose to some other pose in the global frame, we can decompose the motion in two rotations in place and one translation along a straight line. The robot can turn in the place aligning its orientation aiming the goal position, (x_d, y_d), then move forward to the goal position, and then turn in the place again to reach the goal orientation, θ_d.

The image above tries to illustrate the proposed motion. The robot starts with the $\xi_I = [x, y, \theta]^T$. Then it spun around the point P and aim the desired position $P' = (x_d, y_d)$ reaching the pose $\xi'_I = [x, y, \theta_1]^T$. To reach the position, it moves forward to $P' = (x_d, y_d)$ and reaches $\xi''_I = [x_d, y_d, \theta_1]^T$. And then the robot spun again to from θ_1 to θ_d. The final robot state should be $\xi'''_I = [x_d, y_d, \theta_d]^T$.

The Unicycle Model

So far, we saw the kinematics of a two-wheeled robot. But now we talk about a more general and simple model. The previous model tells us how a robot with two wheels can reach a specific pose in the world, acting in the wheel speeds. But, we do not care about how the wheel is turning; we care about the pose of the robot. The unicycle model represents a robot with only one wheel. If the wheel complies with our pure rotation and rolling condition, the wheel has two control inputs, the linear velocity, v, in the axis X_R and the angular velocity, ω, around Z_R. So,

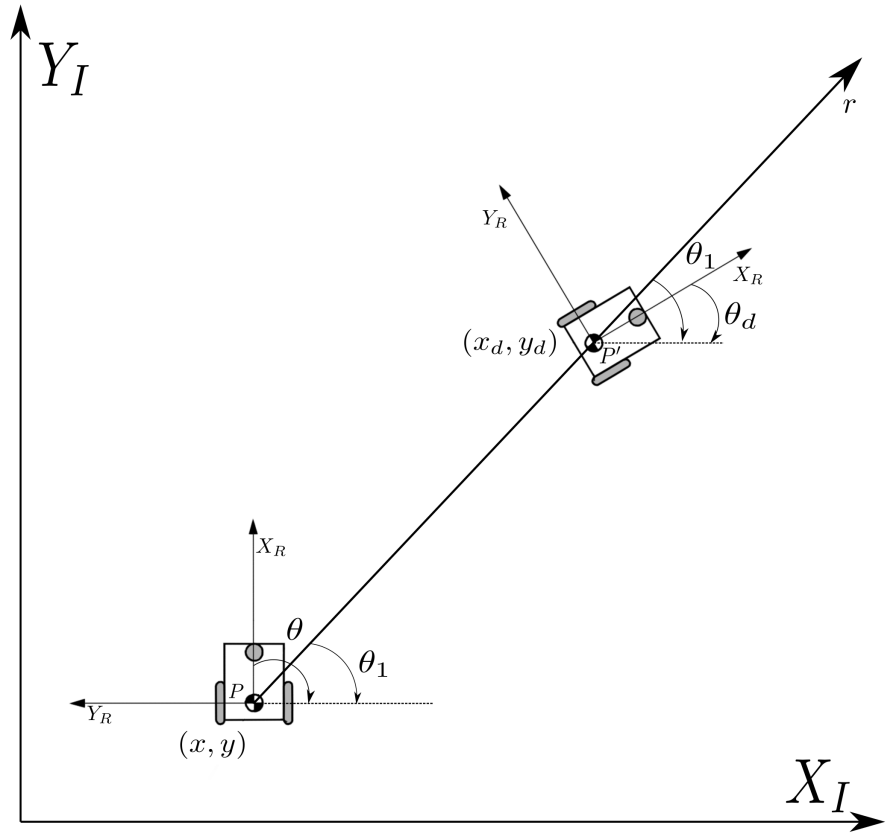


Fig. 5: A robot is moving around with the proposed motion framework.

the kinematics of a unicycle robot described in the inertial frame $\{X_I, Y_I, \}$ is given by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 0 \\ 1 & \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

Where x, y and θ are the coordinates of the robot in the global frame and $u = (v, \omega)$ is the control vector.

Commercial robots usually provide an interface to translate from a desired unicycle control input to desired wheel velocities. And a lower level dedicated microcontroller, which aims to control the wheel velocities.

Notes on Control

So, we should be able to build a system or software capable of, using the maths showed, move a robot to any reachable goal. The control theory is the branch of maths dedicated to this problem. A control system sends inputs to the system and leads the variables of the system to the desired goal. Our system is a mobile robot. And, using the previous equations, the inputs are the spin speed of each wheel, and the output is the pose of the robot.

A controller should give the system the inputs necessary to perform the desired action. As in the image below:

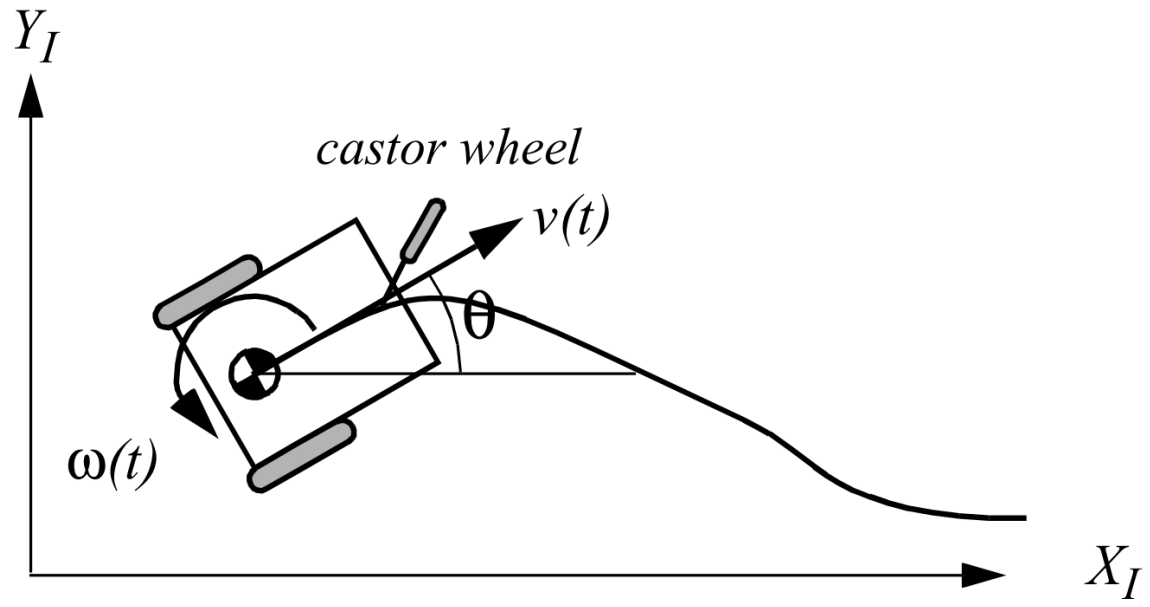
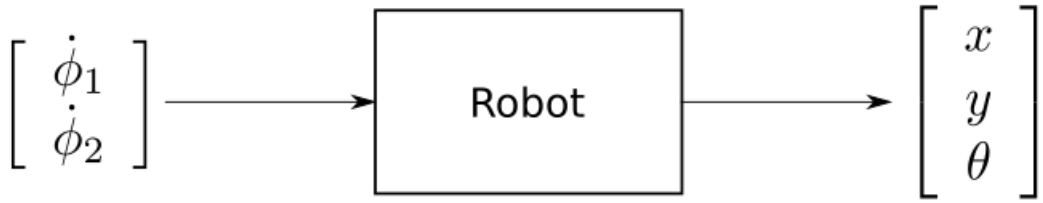
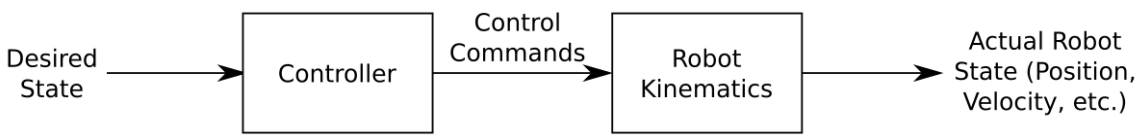
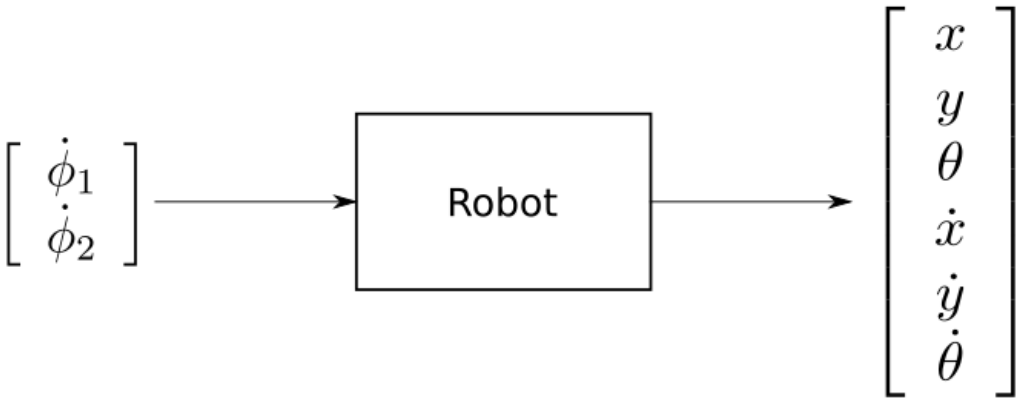


Fig. 6: A differential-drive robot in its global reference frame. Figure from[1].





If we see the controller and the robot as a single system, we can have another system with the desired state as input and the robot state as output. Then we can build a new controller which deals with choosing the desired state. In the same manner, if we would like to control the velocities of the robot and not only the pose, to be able to control how the robot moves. We can add the velocities to the robot state vector and control them with the equations related.



Nota: A differential drive robot has a major problem which is... Feng et al.[4] develops in 1993 a motion controller which...

2.1.2 Probabilistic Kinematics

In the previous section, we try to model the movement of a mobile robot mathematically. As we have seen, we made assumptions to get a simple model capable of describing the movement without loss of generality. Nevertheless, if the assumptions are incorrect, we have to put in the trash all our work? Or, we can make adjustments to continue to describe the actions with the same simplicity of the previous model?

Well, the purpose of this section is to delve into the assumptions. Try to predict when they are incorrect. Furthermore, study a way that could deal with this kind of problem.

In the previous sections, we build a mathematical model looking at a mobile robot modeled with just two wheels. However, our robots have more than two wheels. The P3-DX has three wheels, two motored, and a castor wheel. The P3-AT has four wheels and two motors, one for each side. None of them fits in our model. The castor wheel adds a new problem in the differential drive model. Moreover, the four wheels violate the pure rolling and rotation condition.

The castor wheel problem

The castor wheel, unlike fixed wheels, is an off-centered wheel that is orientable to the robot frame.

The castor wheel does not add any additional constraints to the robot motion, because the robot motion will steer the wheel to a new orientation. However, in a castor wheel, the steering action itself moves the robot chassis because of the offset between the ground contact point and the vertical axis of rotation.

The steering action adds uncertainty to the robot motion. As we can not predict the additional action, we can not guarantee that the control action will drive the robot to the desired pose.

Fig. 7: Castor wheel.

[4] L. Feng, Y. Koren and J. Borenstein, «Cross-coupling motion controller for mobile robots,» in IEEE Control Systems Magazine, vol. 13, no. 6, pp. 35-43, Dec. 1993.

Fig. 8: Castor wheel producing a unpredictable behavior.

The above image shows the simulation of rotation in place of the P3-DX. The castor wheel begins not aligned to the rotation. Then the robot starts to move; this movement steers the castor wheel. Although, the castor wheel pushes the robot away from its original position when steering. We can try to parameterize the steering action, adding a new translation and expand the previous model to add the new change, as we see in[1]. Nonetheless, here we try to generalize this problem as a violation of the motion hypothesis since the robot will not move as the equations tell anymore.

The assumption violation

Velocity Motion Model

Odometry Motion Model

2.1.3 Robots

Nota: All robot's name were referencies to The Three Musketeers

Athos

Athos is Pioneer 3-AT

Aramis

Aramis is Pioneer 3-DX

Porthos

Porthos is Pioneer 3-AT

2.1.4 Computer

All the three robots have the same onboard computer, to run its system.

Computer Specs

- Model IPX1800G2
- Processor Intel® Celeron Dual-Core J1800, 2.41GHz
- RAM 4 GB DDR3 SODIMM
- SSD 120 GB
- 6 USB 2.0
- 1 USB 3.0

[1] Roland Siegwart and Illah R. Nourbakhsh. 2004. Introduction to Autonomous Mobile Robots. Bradford Company, USA.

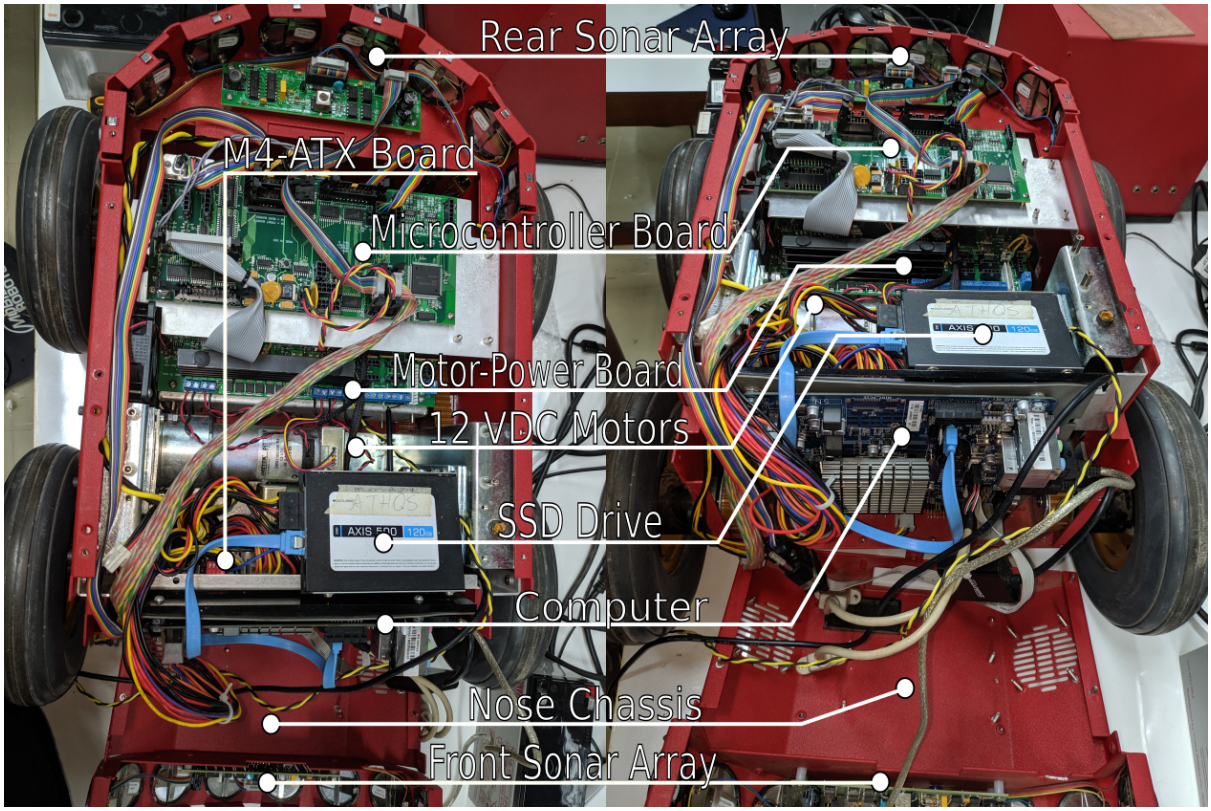


Fig. 9: Athos inside.

- 1 RS232, already used for robot's microcontroller
- WiFi
- VGA and HDMI out

Power Supply

The Computer is powered by a DC-DC Circuit. Athos and Aramis have an M4-ATX and Porthos have a Pico PSU-160-XT. They are only used to power up the computer using the 24 pin ATX connector. Others accessories, such as Kinect and cameras, use the Motor Power Board.

> **Atenção:** The Pioneer family has a board which all the batteries are connected, there is a 20 Amp car fuse in this board. Please be sure that this fuse is connected and working. The robot should not be able to power up without it.

M4-ATX

The M4-ATX Power Supply is manufactured by Minibox.

- M4 ATX Manual

PSU-160-XT Specs

- PSU 160 XT Manual

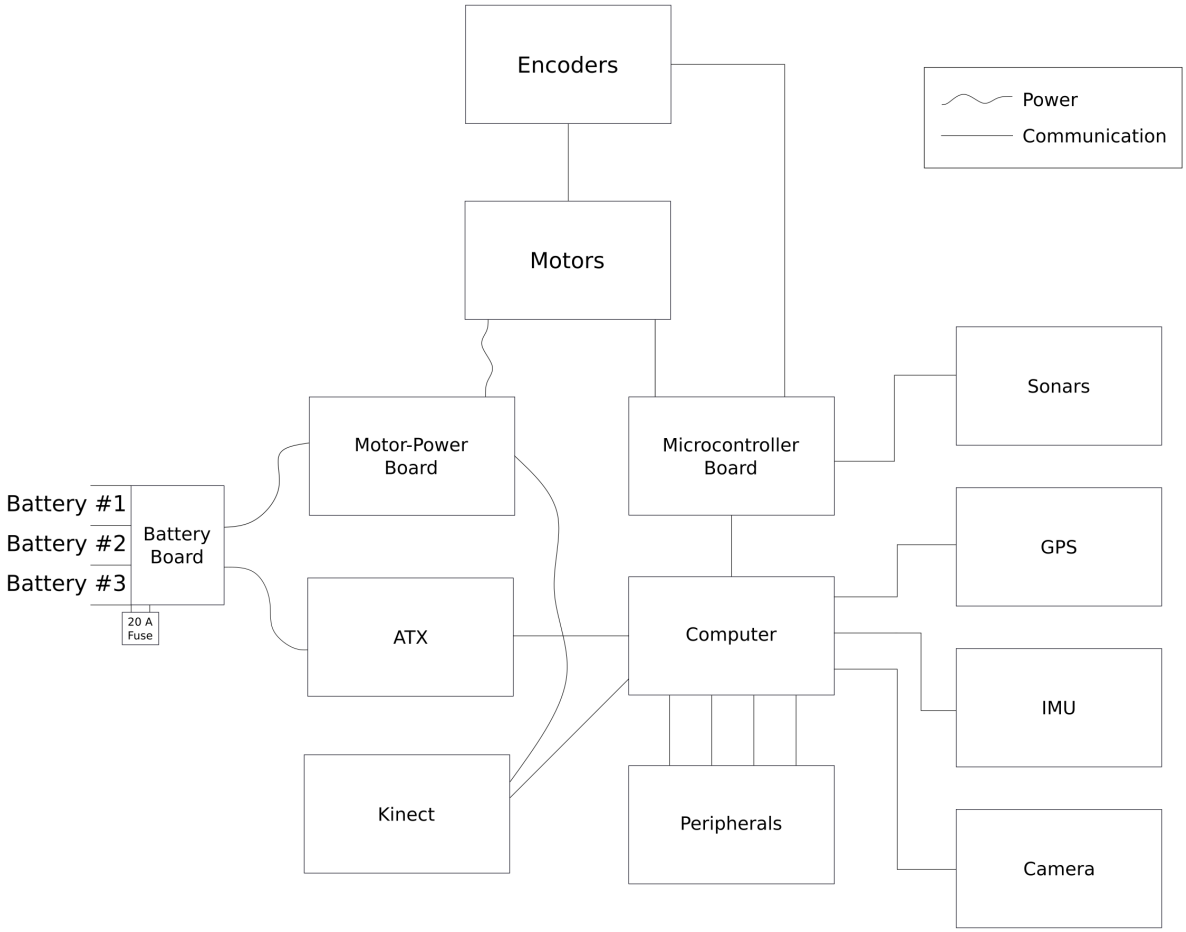


Fig. 10: Hardware Diagram.

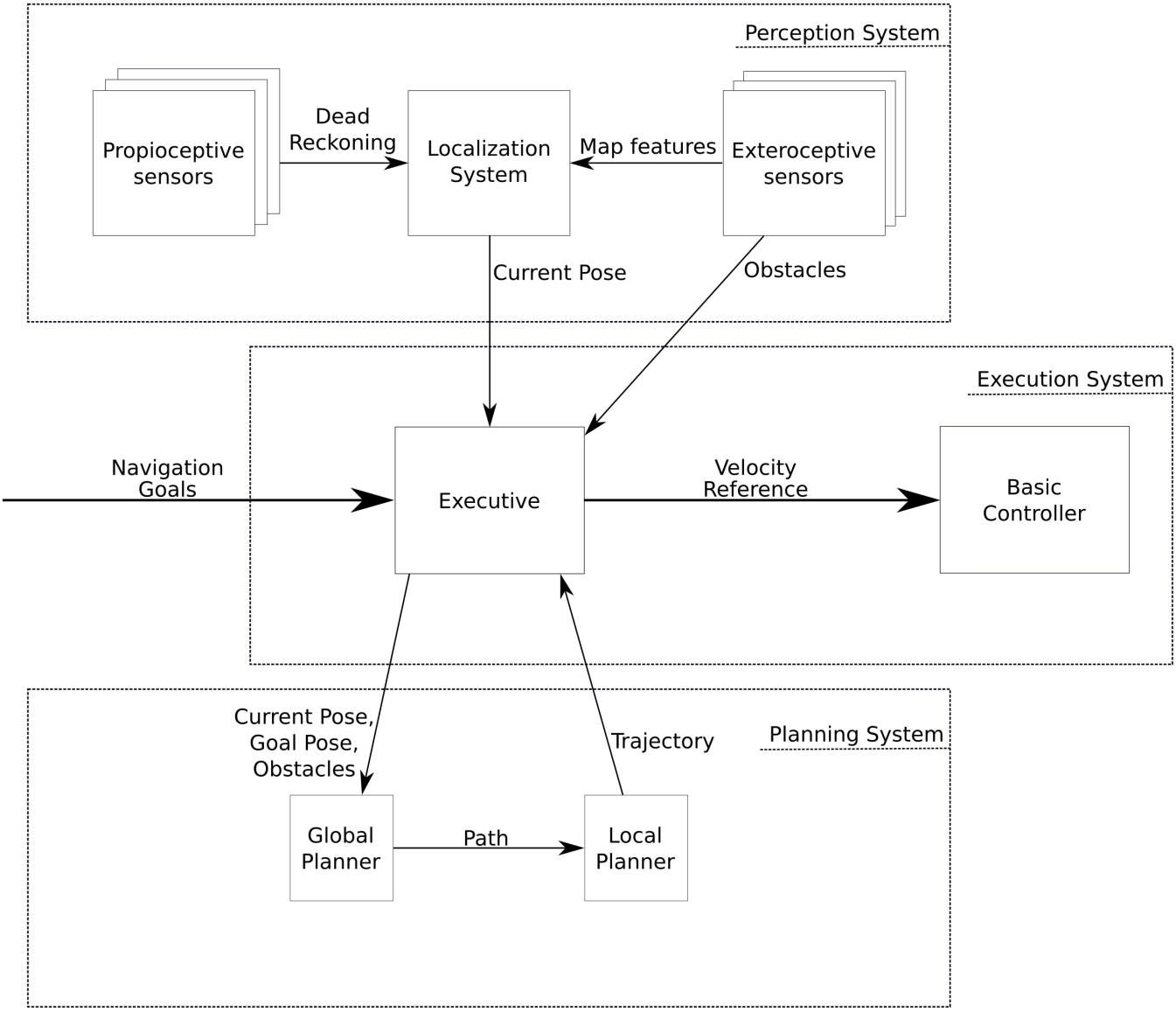


Fig. 11: Navigation System Diagram.

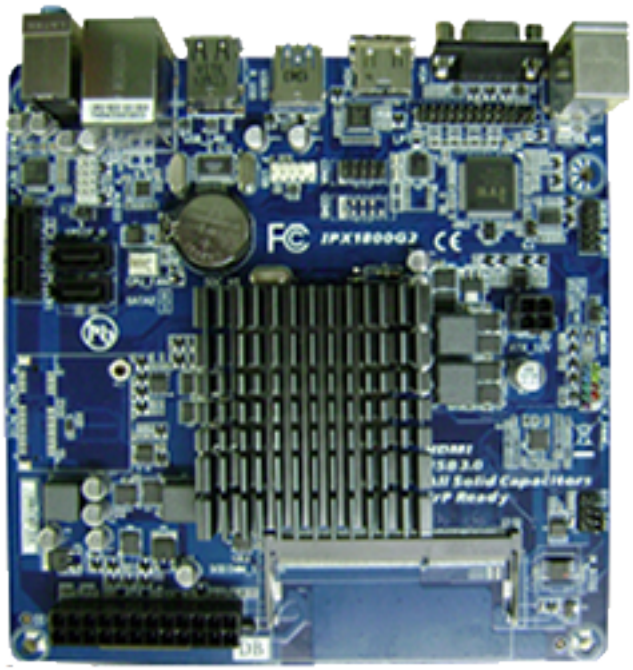
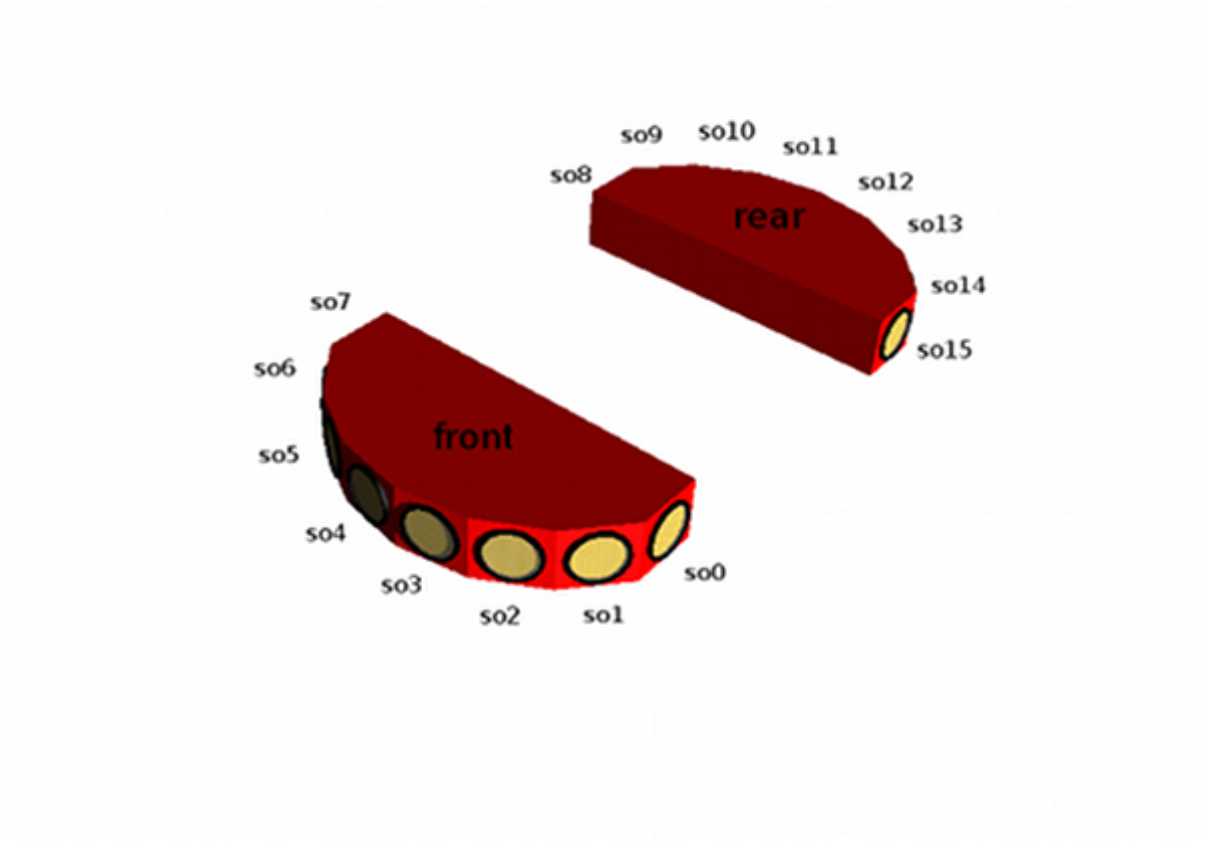


Fig. 12: IPX1800G2 from PCWARE

2.1.5 Sonars

Athos and Porthos has two sonar arrays (front and rear), while Aramis have only one (front) sonar array, sonar or ultrasoud is a sensor that uses sound wave for detect obstacles and range information for collision avoidance.



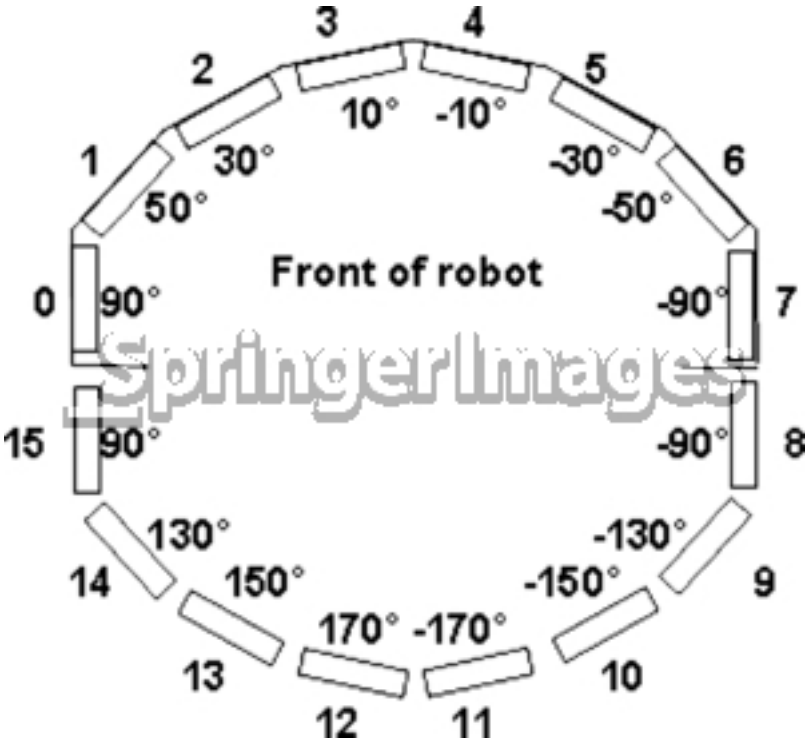
Sonar Specs

- Range of view: 0.1 m ~ 5 m
- Aquisition rate: 25 Hz

Aviso: If the sonar doesn't view anything in its cone of view, it will send to the software the max range.

Geometry

The position of each sonar is showed in the image below.



Importante:

- All these locations are fixed in the robot.
- There're a URDF file, describing these locations to ROS. Please see the section description of the robot.

Sensitivity Adjustment

The driver electronics for each array is calibrated at the factory. However, you may adjust the array's sensitivity and range to accommodate differing operating environments. The sonar gain control is on the underside of the sonar driver board, which is attached to the floor of each sonar module.

Sonar sensitivity adjustment controls are accessible directly, although you may need to remove the Gripperto access the front sonar, if you have that accessory attached. For the front sonar, for instance, locate ahole near the front underside of the array through which you can see the cap of the sonar-gain adjustment potentiometer. Using a small flat-blade screwdriver, turn the gain control counterclockwise to make the sonar less sensitive to external noise and false echoes.

Low sonar-gain settings reduce the robot's ability to see small objects. Under some circumstances, that is desirable. For instance, attenuate the sonar if you are operating in a noisy environment or on uneven or highly reflective floor,

a heavy shag carpet, for example. If the sonar are too sensitive, they will "see" the carpet immediately ahead of the robot as an obstacle.

Increase the sensitivity of the sonar by turning the gain-adjustment screw clockwise, making them more likely to see small objects or objects at a greater distance. For instance, increase the gain if youare operating in a relatively quiet and open environment with a smooth floor surface.

By Mobile Robots©

Dica: See more in the manual.

Software

We use the p2os to read the sonars readings and the sonar_viz to transform the readings in standart data to better maniplation.

2.1.6 Cameras

Athos have a Stereo Camera, Aramis a Kinect and a USB Camera, Porthos only Kinect.

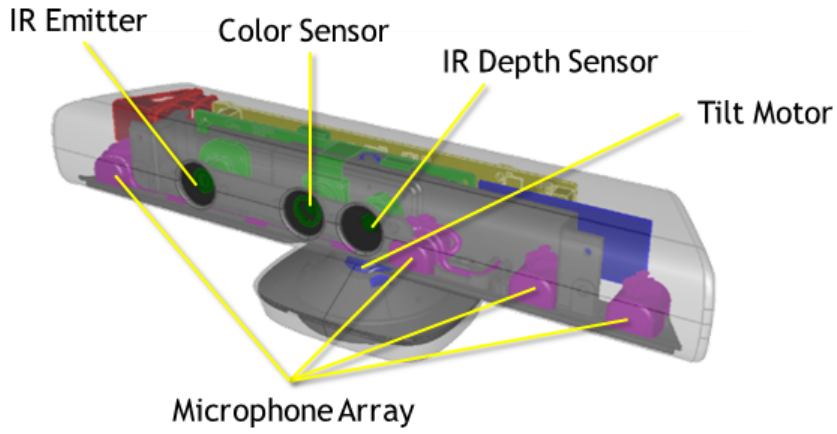
Stereo Cameras

Specs

- Aquisition rate: 30 fps

Kinect

Kinect is a motion sensor device by Microsoft©



Specs

- Color Camera Resolution 640 x 480
- Depth Camera Resolution 320 x 240
- Max Depth Distance ~4.5 m
- Min Depth Distance ~0.04 m
- Viewing angle 43° vertical by 57° horizontal field of view
- Vertical tilt range +-27°
- Frame rate (depth and color stream) 30 fps
- Audio format 16-kHz, 24-bit mono pulse code modulation (PCM)
- Audio input characteristics A four-microphone array with 24-bit analog-to-digital converter (ADC) and Kinect-resident signal processing including acoustic echo cancellation and noise suppression
- Accelerometer characteristics A 2G/4G/8G accelerometer configured for the 2G range, with a 1° accuracy upper limit.

USB Camera

The USB Camera in Aramis is a FMVU-03MTM-CS from Point Grey.

Specs

- Resolution 752 x 480
- Frame Rate 60 fps
- Megapixels 0.3 MP
- Chroma Mono
- Sensor Type CMOS

- Readout Method Global shutter
- Interface USB 2.0

2.1.7 Batteries

All the three musketeers contain three sealed lead-acid batteries accessible through a hinged and latched rear door. The batteries charge life typically ranges from two to three hours.

Importante: Batteries have a significant impact on the balance and operation of your robot. Under most conditions, we recommend operating with three batteries. Otherwise, a single battery should be mounted in the center, or two batteries inserted on each side of the battery container.

By Mobile Robots©

Batteries Specs

- Lead-acid
- Sealed
- 12 VDC
- 4 Ah
- With 3 batteries, 252 Wh
- Hot-swappable

Battery Indicators and Low Voltage Conditions

The User Control Panel[1] has a bi-color LED labeled BATTERY that visually indicates current battery voltage. From approximately 12.5 volts and above, the LED glows bright green. The LED turns progressively orange and then red as the voltage drops to approximately 11.5 volts.

Arually, the buzzer will sound a repetitive alarm if the battery voltage drops below 11.5 VDC. If the battery voltage drops below 11 VDC the microcontroller automatically shuts down a client connection and notifies the computer to shut down.

Nota: The batteries voltage is monitored by a own package, this package if necessary notifies the user and shut down automatically the operating system. See more in robot monitor.

Atenção: The Pioneer family has a board which all the batteries are connected, there is a 20 Amp car fuse in this board. Please be sure that this fuse is connected and working. The robot should not be able to power up without it.

Recharging

Standart Charger

This accessory recharge the batteries in the fast-charge mode (4A maximum current). The fast-charge mode is showed with an orange LED and trickle mode by a green LED, which the batteries are given only enough current to remain at full charge.

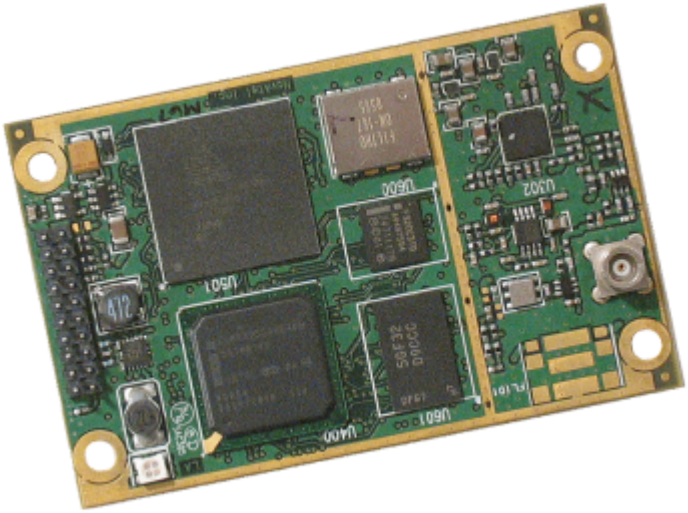
[1] See in the manual.

> **Aviso:** In the fast-charge mode, care must be taken to charge at least two batteries at once. A single battery may overcharge and thereby damage both itself and the robot.

Power Cube

This accessory allows simultaneous recharge of three batteries outside the robot.

2.1.8 GPS Receiver



The robots have each one a Novatel GPS receiver model OEMV-1. We have a custom software to publish the GPS readings in ROS and a custom hardware to power the receiver.

Specs

- GPS tracking
- L1, L-Band and SBAS signal tracking
- Low power consumption for longer operating time
- Single frequency

Antenna

«The ANT-35C1GA-TW-N is an active GPS antenna, 88.9 mm (3.5”) in diameter, and designed to operate at the GPS L1 frequency of 1575.42 MHz. Its mechanical configuration is a spherical radius molded radome which provides enhanced protection against rain and ice.»[1]

Power Board

A circuit board was built to integrate GPS and IMU readings, to communicate with onboard pc and to power the GPS receiver and the IMU.

2.1.9 IMU

[1] See more in the antenna user guide.

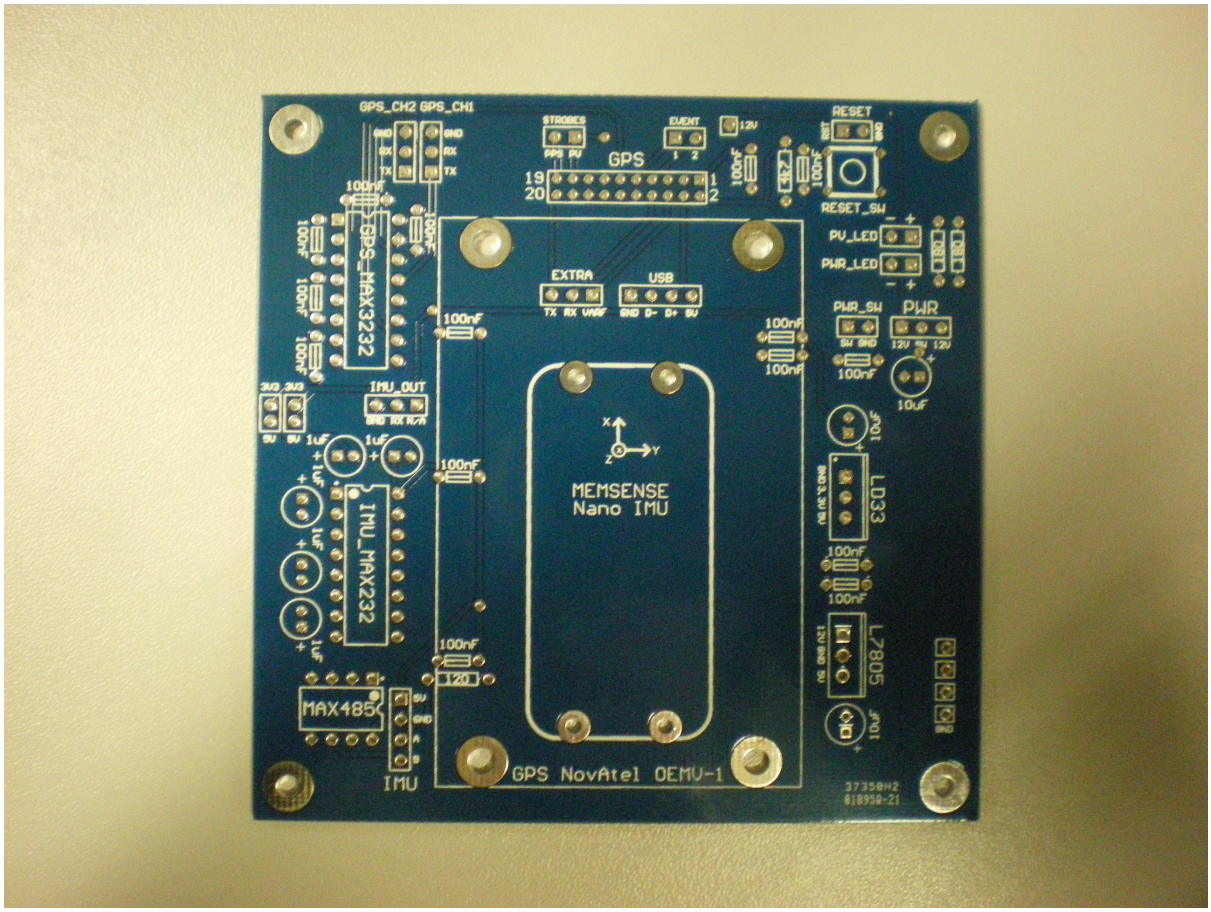


Fig. 13: Power Board

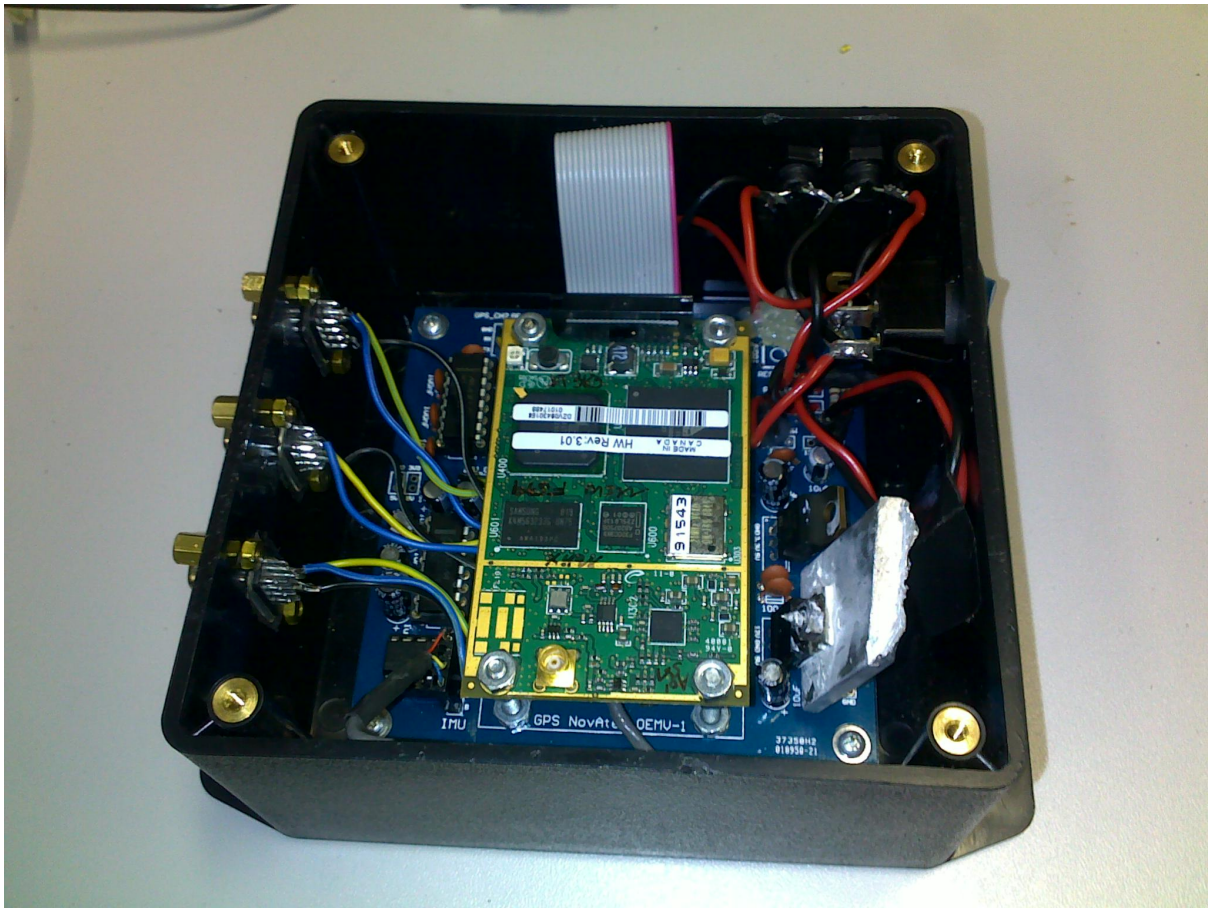


Fig. 14: Power Board with GPS Receiver and IMU (under GPS) in its case.

The MEMSense NanoIMU is a 9DOF IMU with thermometer used in the robots together with GPS receiver. We have a custom software and a hardware to power and send the readings to the computer.

Specs

Gyrometer Specs

- Dynamic Range +-300 º/s
- Offset +-1.5 º/s
- Cross-axis sensitivity +-1 %
- Nonlinearity +-0.1 % of FS (Best fit straight line)
- Noise 0.56 (max 0.95) º/s, sigma
- Digital Sensitivity 1.3733E-2
- Bandwidth 50 Hz

Accelerometer Specs

- Dynamic Range +-2 g
- Offset +-30 mg
- Nonlinearity +-0.4 (max +-1.0) % of FS
- Noise 0.6 (max 0.8) mg, sigma
- Digital Sensitivity 9.1553E-5
- Bandwidth 50 Hz

Magnetometer Specs

- Dynamic Range +-1.9 gauss
- Drift 2700 ppm/ºC
- Nonlinearity +-0.5 % of FS (Best fit straight line)
- Noise 0.00056 (max 0.0015) gauss, sigma
- Digital Sensitivity 8.6975E-5
- Bandwidth 50 Hz

Thermometer Specs

- Digital Sensitivity 1.8165E-2

Power Board

The circuit board is the same for GPS receiver, more informations here.

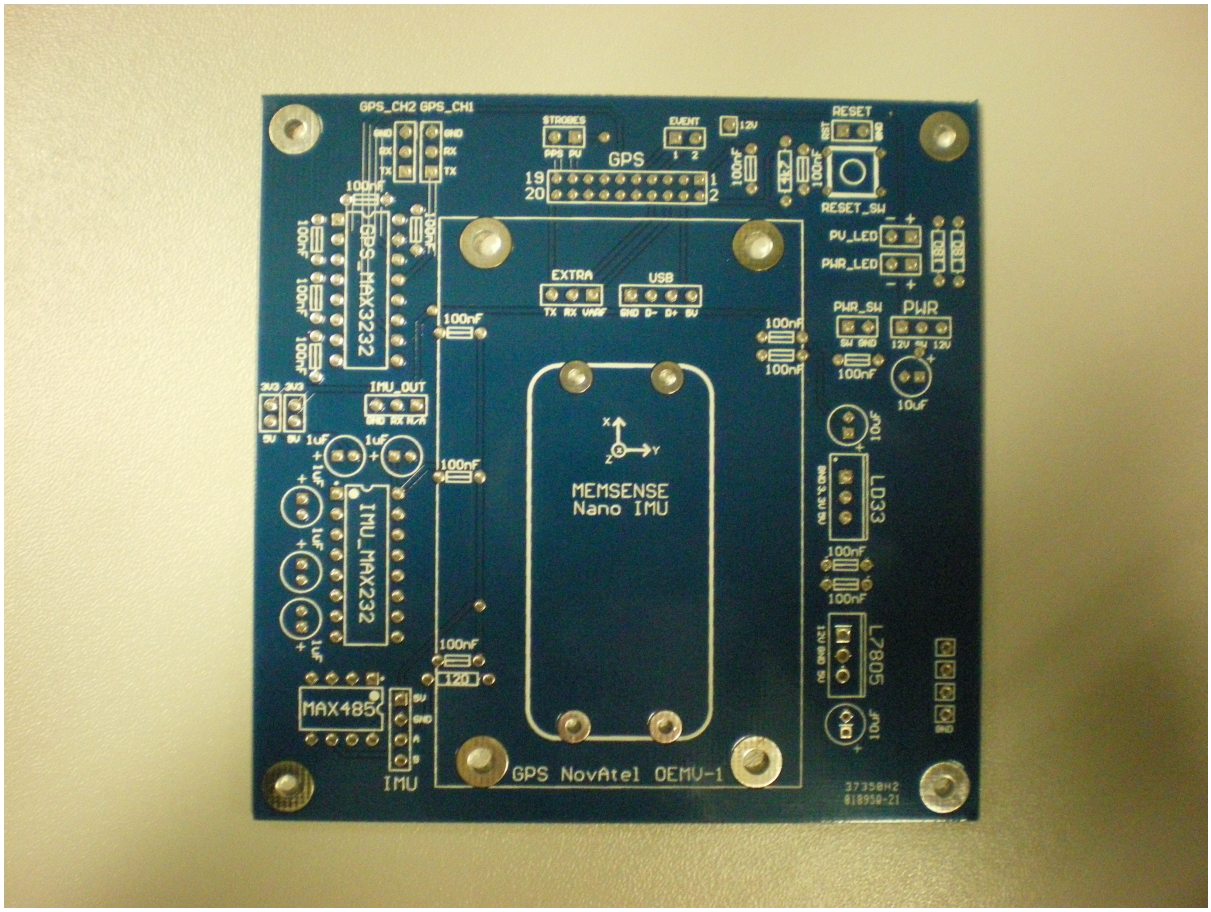


Fig. 15: Power Board

2.1.10 ROS

2.1.11 Gazebo Simulator

2.1.12 P2OS Package

2.1.13 Move Base Package

2.1.14 System's Architecture

2.1.15 Mobile Robotics Lab 1

Objetivos

Aprender a criar um pacote no ROS.

Introdução

Todo o software produzido no ROS é organizado em pacotes. Um pacote pode conter nós do ROS, bibliotecas independentes, datasets, arquivos de configuração ou qualquer material que possa constituir um módulo. Para que um pacote seja visto pelo ROS ele deve ter no mínimo os seguintes requisitos:

- Deve conter um arquivo chamado package.xml com código catkin;
- Deve conter um arquivo CMakeLists.txt;
- Cada pacote deve ter seu próprio diretório;

O arquivo package.xml reúne todas as informações de autor, nome do pacote, versão do código e dependências. Já o CMakeLists.txt reúne as regras de compilação do pacote.

Procedimentos

- **No terminal, configure o ros com o comando:**
  - $ bash.sh
- **No novo terminal aberto, entre na pasta catkin_ws. Esse será o ambiente de trabalho a ser usado neste curso:**
  - $ cd catkin_ws
- **Entre na pasta src. Para o ROS essa é pasta onde fica o código fonte dos pacotes:**
  - $ cd src
- **Use o script catkin_create_pkg para criar um novo pacote. Esse pacote será chamado de 'beginner_tutorials' e terá co**
  - $ catkin_create_pkg beginner_tutorials std_msgs rospy roscpp
- **Este comando irá criar uma pasta chamada de beginner_tutorials que conterá um package.xml e um CMakeLists.txt**
  - Abra o CMakeLists.txt em um editor de texto e procure onde as dependências são chamadas.
  - Abra o package.xml em um editor de texto e procure onde as informações de autor, versão e dependências são informadas.
- **Para compilar o workspace volte para catkin_ws:**
  - $ cd ~/catkin_ws
- **Atualize as variáveis de ambiente do ROS:**

  - $ cd catkin_ws
  - $. devel/ros_custom.sh
- **Compile o workspace:**
  - $ catkin_make

Exemplo

Referências

Esse tutorial foi baseado no tutorial Creating a ROS Package do ROS:

http://wiki.ros.org/ROS/Tutorials/CreatingPackage

2.1.16 Mobile Robotics Lab 2

Objetivos

Aprender sobre mensagens e tópicos do ROS.

Procedimentos

Abra o terminal e inicie o nó central do ROS:

- $ roscore

Em outro terminal inicie o turtlesim:

- $ rosrun turtlesim turtlesim_node

O ROS usa um sistema de comunicação distribuída em que os executáveis, chamados de nós, usam tópicos para a troca de n

- $ rostopic list

Este comando irá listar todos os tópicos ativos. Para ver o nós ativos use:

- $ rosnode list

Para ver as informações de um nó, use rosnode info /nome_do_no:

- $ rosnode info /turtlesim

Este comando irá mostrar os tópicos que são publicados e ouvidos pelo nó turtlesim. Para mover a tartaruga do turtlesim d

- $ rostopic info /turtle1/cmd_vel

Este comando mostrará o tipo de mensagem, os 'publishers' e os 'subscribers' deste tópico. O tipo 'geometry_msgs/Twist' r

- $ rosmsg show geometry_msgs/Twist
- Obs.: Todas as unidades no ROS estão no Sistema Internacional de Unidades (SI), portanto a velocidade se dá em m/s;

Para publicar uma mensagem no tópico /turtle1/cmd_vel, use o comando:

- $ rostopic pub /turtle1/cmd_vel geometry_msgs/Twist «{linear:[0.0, 0.0, 0.0], angular:[0.0, 0.0, 0.0]}»
- Obs.: a tecla TAB pode ser usada para completar os comandos;

O primeiro argumento da mensagem representa a velocidade linear e suas componentes x, y e z. Já o segundo argumento re

- $ rostopic pub /turtle1/cmd_vel geometry_msgs/Twist «{linear:[1.0, 0.0, 0.0], angular:[0.0, 0.0, 0.0]}»

Para enviar uma mensagem em loop, deve-se adicionar o argumento '-r <frequência_do_loop>'. Para enviar o mesmo coma

- $ rostopic pub /turtle1/cmd_vel geometry_msgs/Twist «{linear:[0.0, 0.0, 0.0], angular:[0.0, 0.0, 1.0]}» -r 10

Para ver o que está sendo publicado em um nó pode se usar 'rostopic echo':

- $ rostopic echo /turtle1/cmd_vel
- $ rostopic echo /turtle1/pose

Usando os comando acima faça a tartaruga desenhar uma linha. Usando os comando acima faça a tartaruga desenhar um quadrado. Usando os comando acima faça a tartaruga desenhar um círculo.

Referências

Para mais informações sobre tópicos, mensagens e nós visite:

- http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes
- http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics
- Referência sobre geometry_msgs: http://wiki.ros.org/geometry_msgs
- Referência sobre o turtlesim: http://wiki.ros.org/turtlesim

2.1.17 Mobile Robotics Lab 3

Objetivos

O objetivo desse exercício é programar a movimentação de um robô diferencial usando os comando no terminal aprendidos no laboratório passado. Este exercício usará uma versão simulada do robô Pioneer.

Procedimentos

Na pasta src do seu workspace (catkin_ws/src/) baixe o pacote fcr2018

- $ git clone https://github.com/Gastd/fcr2018

Atualize as variáveis de ambiente do ROS

- $ cd ~/catkin_ws
- $ source devel/ros_custom.sh

Abra a simulação do pioneer

- $ roslaunch fcr2017 pioneer3at.gazebo.launch

Use o 'rostopic list' para descobrir quais mensagens o pioneer publica e quais ele recebe. Use o 'rostopic echo' para olhá-las.

Descubra qual tópico o robô usa para publicar as informações do laser

Descubra qual tópico o robô usa para publicar as informações da odometria

Descubra qual tópico é usado para enviar velocidades para o robô

Use o 'rostopic pub' para publicar velocidades. Escolha números que façam o robô andar em uma linha reta

Faça o robô andar em um círculo

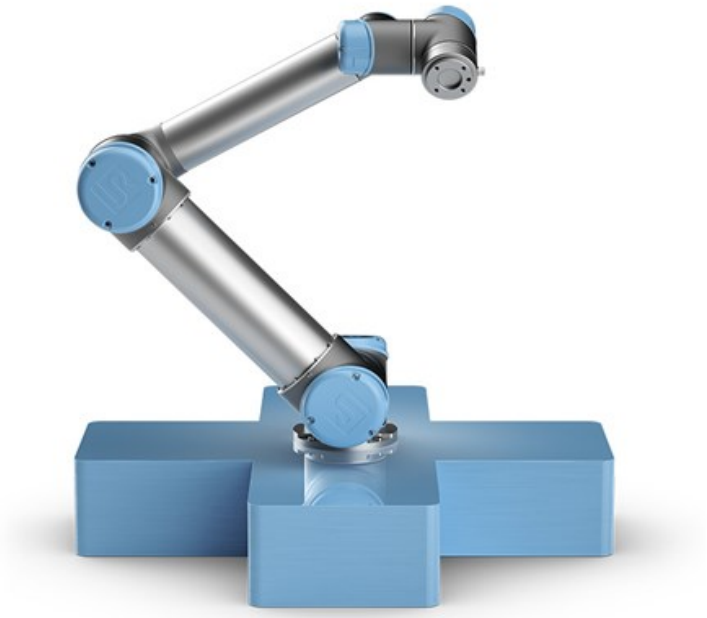
Faça o robô andar em um retângulo

Referências

CAPÍTULO 3

Manipulators

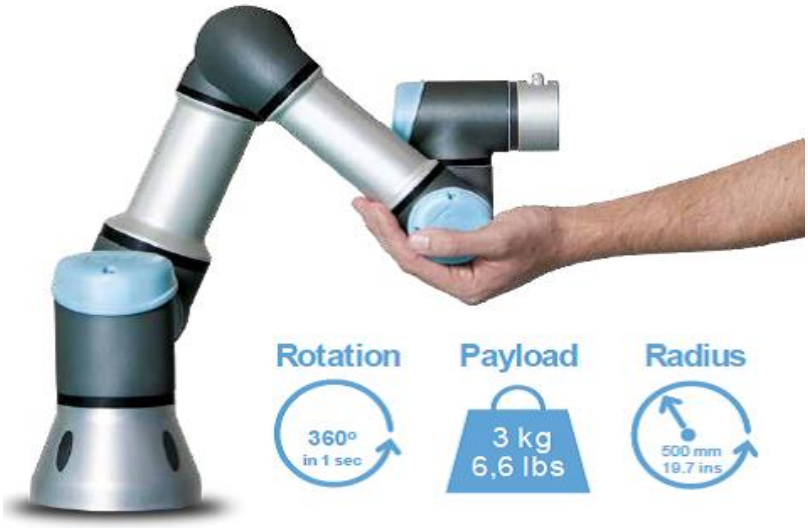
3.1 UR3



3.1.1 System Description

Overview

The UR3 is a table-top collaborative robot. With its 3 kg payload it is very capable and its small footprint makes it suitable for limited workspace situations. With its infinite turn on the end joint, several activities can be perfomed



with grippers attached at robot tool connector.

Some of its applications:

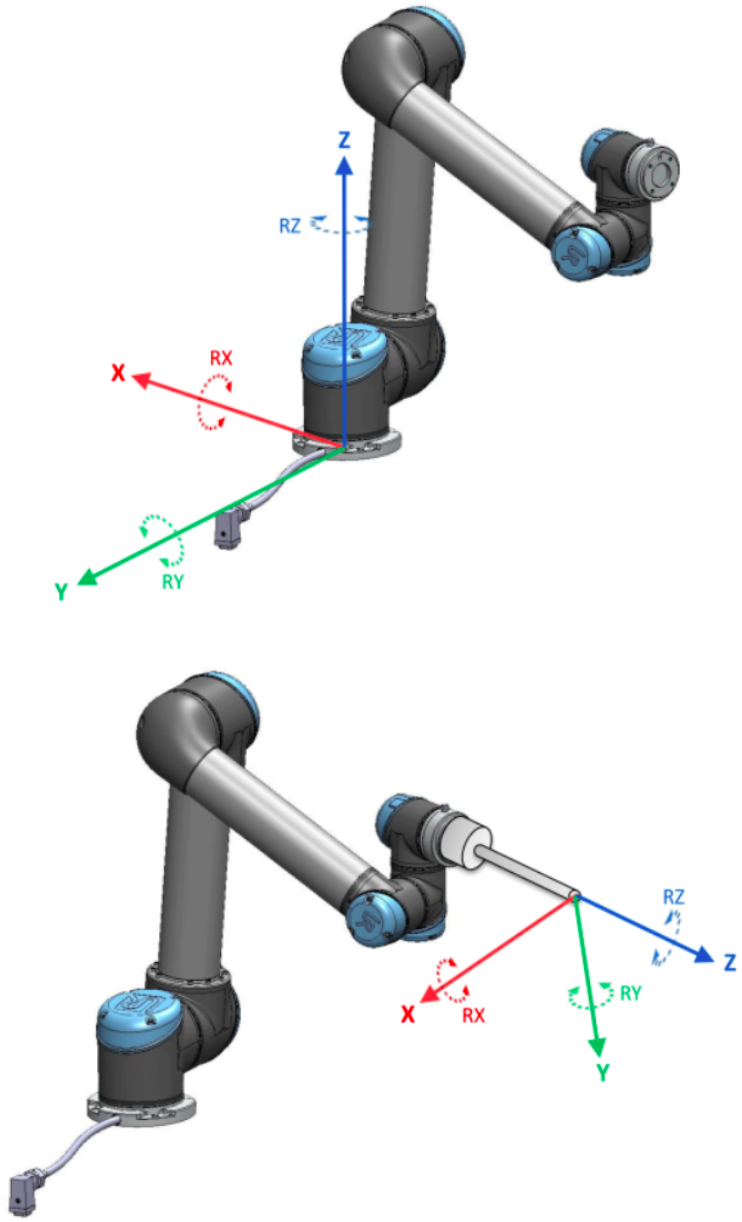
- Laboratory work
- Assembly tasks
- Polishing
- Soldering
- Gluing
- Screwing
- Painting
- Pick and place
- Operating hand tools
- Fume hood tasks

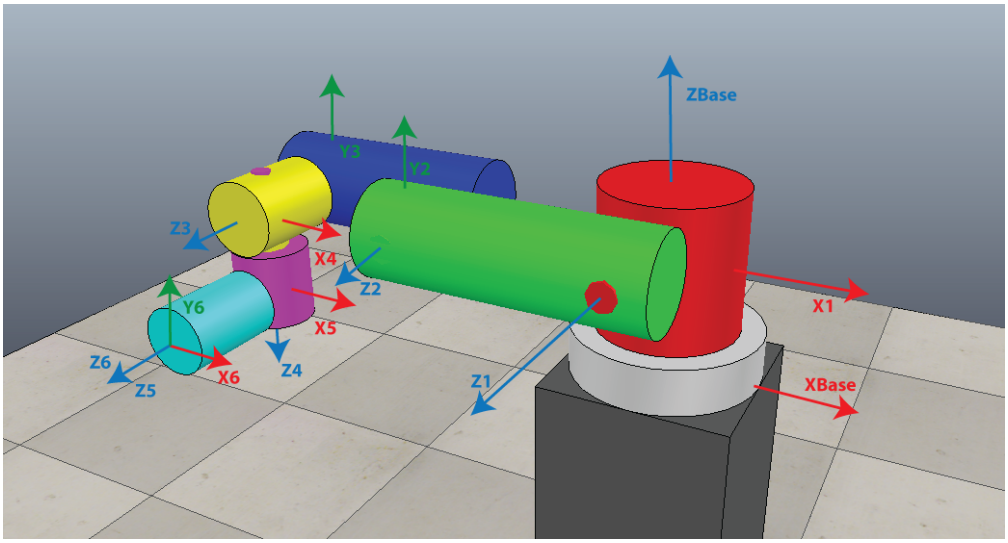
Specification

Table 1: UR3 Specification

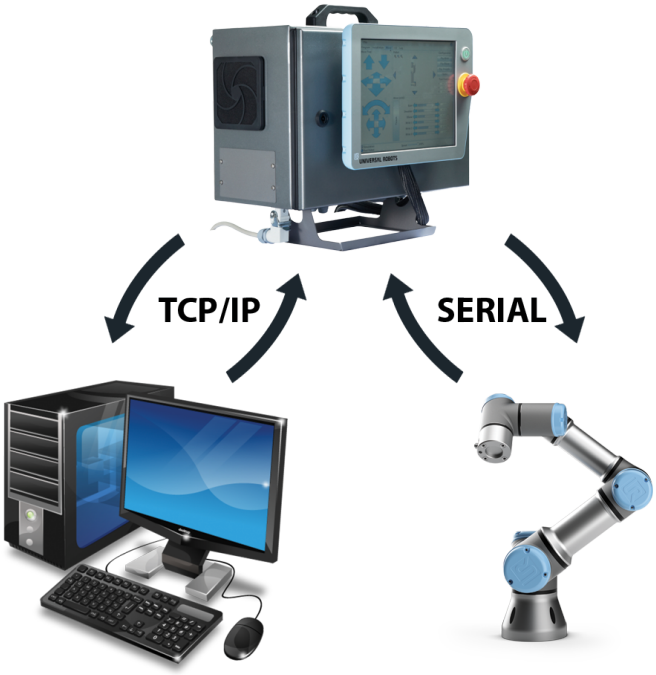
Weight	11.2 kg
Payload	3 kg
Reach	500 mm
Footprint	Ø 128 mm
Degrees of freedom	6 rotating joints
Joint ranges	+/- 360°, infinite rotation on end joint
Speed wrist joints	360 degrees/sec
Other joints	180 degrees/sec
Noise	Comparatively noiseless
IP classification	IP64

Coordinate System




ZBase
Y3
Y2
Z3
X4
Z2
X1
Y6
X5
Z6
Z4
Z5
X6
Z1
XBase

Communication



There are two main communication gateway in this system.

Foremost, there is a TCP/IP communication port between a LinuxPC and the UR3 Control Box. Basically, it is possible to send ROS commands directly from from Linux Terminal or specialized simulation softwares.

Next, there is a serial communication port between the Control Box and the UR3 Robot. It is responsible for send all the position and velocity commands to the robot.

Some of its specification:

- TCP/IP 100 Mbit: IEEE 802.3u, 100BASE-TX
- Ethernet socket & Modbus TCP

Control Box

Linux PC

3.1.2 Kinematics

3.1.3 Dynamics

3.1.4 Robots

3.1.5 ROS

3.1.6 Lab 1: First Contact

objective

The new user contacts the ur3 platform and performs a first experiment.

Introduction

This lab teaches how to run a code to make a simple move on the ur3 robot using ROS.

Procedures to setup the ur3 robot

First, let's turn on the UR3 robot by pressing the POWER button shown in figure 1.

Fig. 1: Figure 1: Power button

After pressing the power button, the screen like the figure below will appear.

Fig. 2: Figure 2: Loading screen

After loading the robot system, the screen as the figure below will appear.

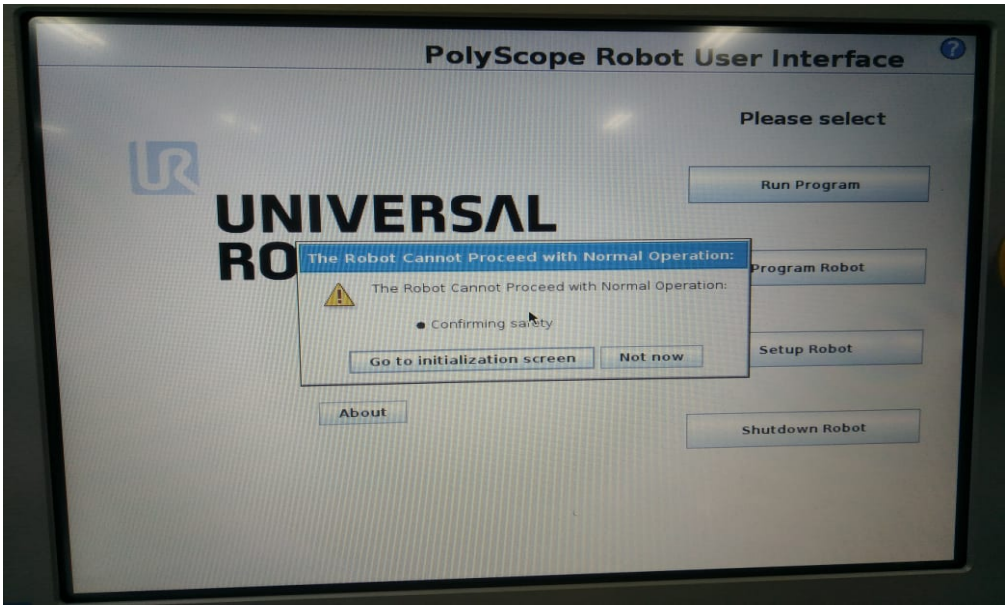


Fig. 3: Figure 3: Apresentation screen

Now, turn on the control box by clicking **Go to the initialization screen**, which is in the center of figure 3.

now, we have a screen like figure 4.

The turn on the robot, press the **ON** button. The screen that will appear will be like as in figure 5.

The robot is now on. To unlock the joints, press the **START** button. The robot will make an unlocking sound, which is expected, and will enter the normal operating mode that can be seen in figure 6.

Depois disso, aperte **OK** que fica localizado no canto inferior direito da tela (figura 6). Now, a new screen will appear (figure 7) with some options for robot functionality. Let's click on **Program Robot**.

Depois de apertar **Program Robot** aparecerá uma tela semelhante a figura 8. Aperte **Move**.

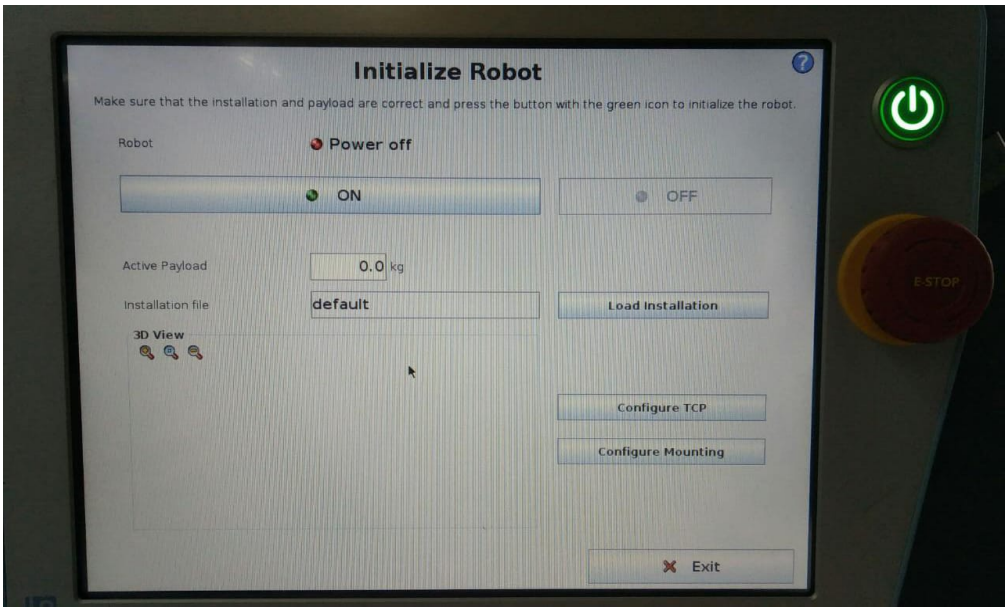


Fig. 4: Figure 4: screen to turn on the robotic arm.

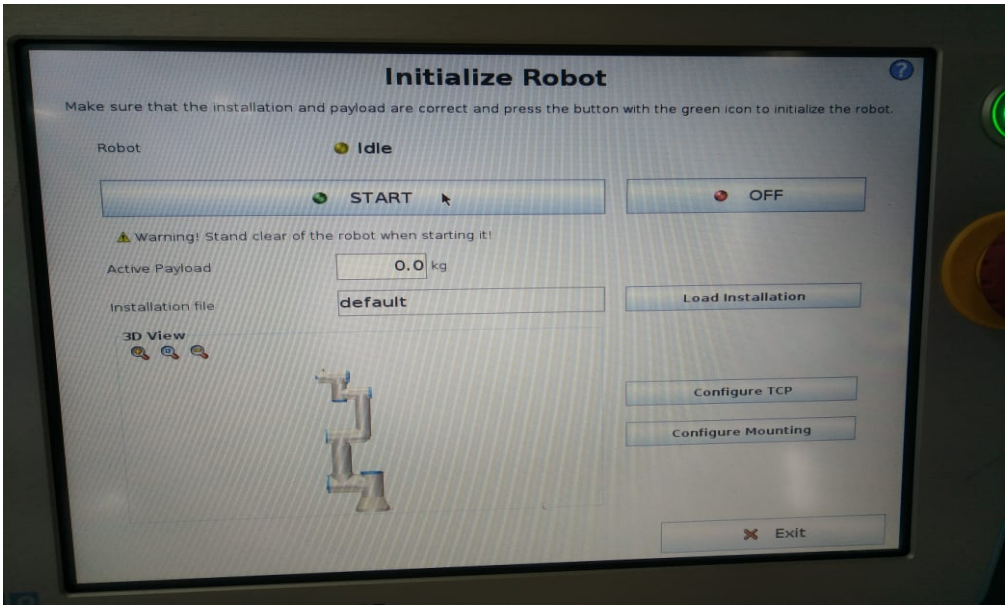


Fig. 5: Figure 5: start screen.

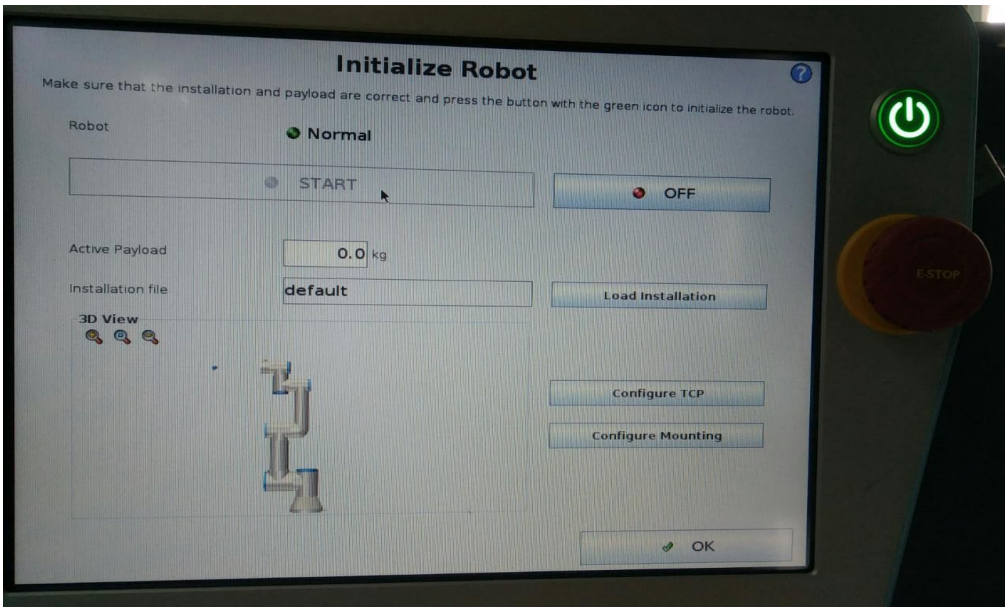


Fig. 6: Figure 6.

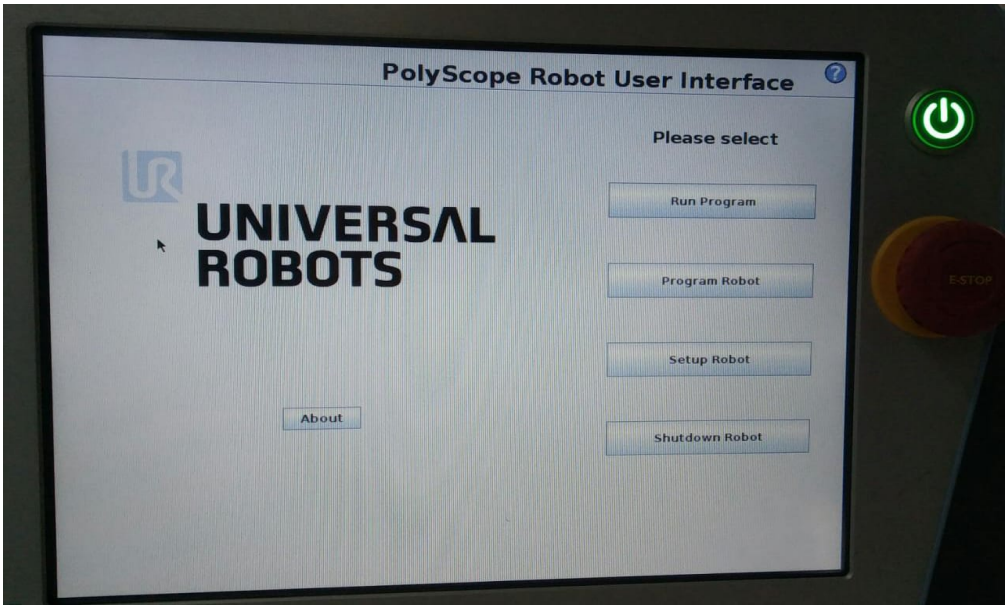


Fig. 7: Figure 7.

Ready. We have completed our robot setup and the final screen is similar to Figure 9.

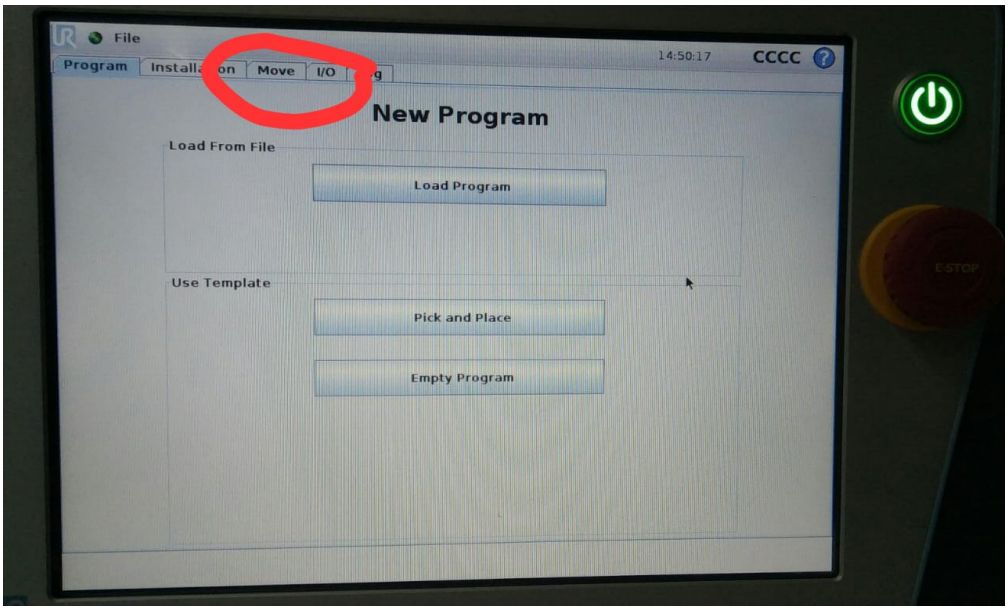


Fig. 8: Figure 8.

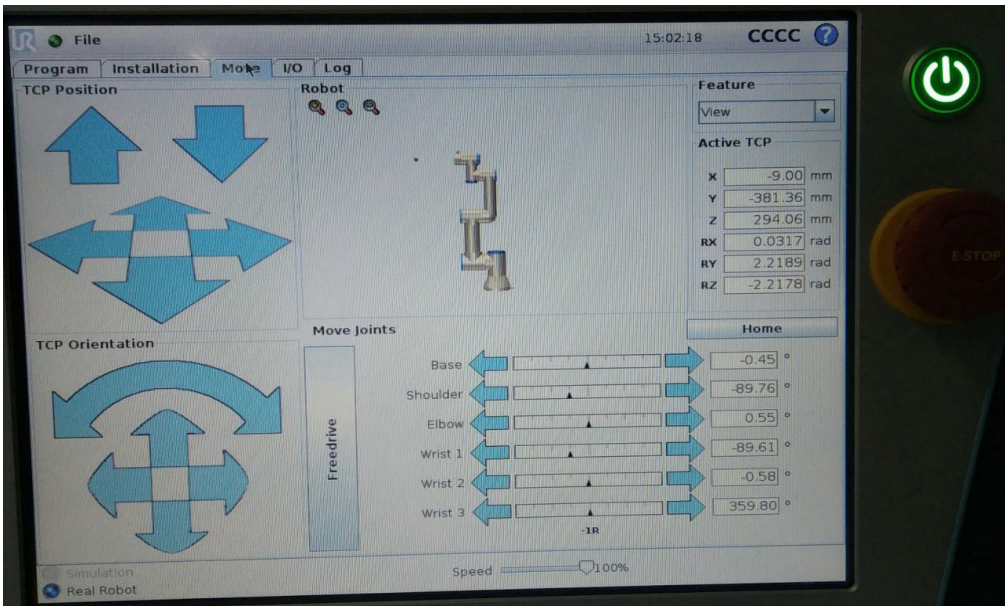


Fig. 9: Figure 9.

Procedures to setup the Linux PC

Now let's turn on the computer that controls the robot. To do this, connect the LARA computer designated for the UR3 robot. The computer name is **ur3** and password **ur3**.

One more thing, the communication will be done using the TCP/IP protocol using cable, so check if the network cable of the Lara-robots router is connected to the ur3 computer.

After making the instructions described in the paragraph above, we will download the files that contain the codes for our experiment. To do this, open the application called **Terminator** on the ubuntu 18.04 taskbar, which is exactly like figure 10.

Fig. 10: Figure 10: Terminator

After opening **Terminator**, we will use the command **git clone** to download the file. Then, type in the Terminator the command (If the file already exists, skip this step):

- $ git clone https://github.com/lara-unb/UR3_interface.git

The next step is to compile the code to «run» on the ur3 computer. To compile the code we will enter the directory, using Terminator, where the executable folders are.

With the command shown below you can enter the interface code workspace.

- $ cd UR3_interface/catkin_ur3

To compile the code use the command:

- $ catkin_make

You will receive a message like the one in figure 11 below.

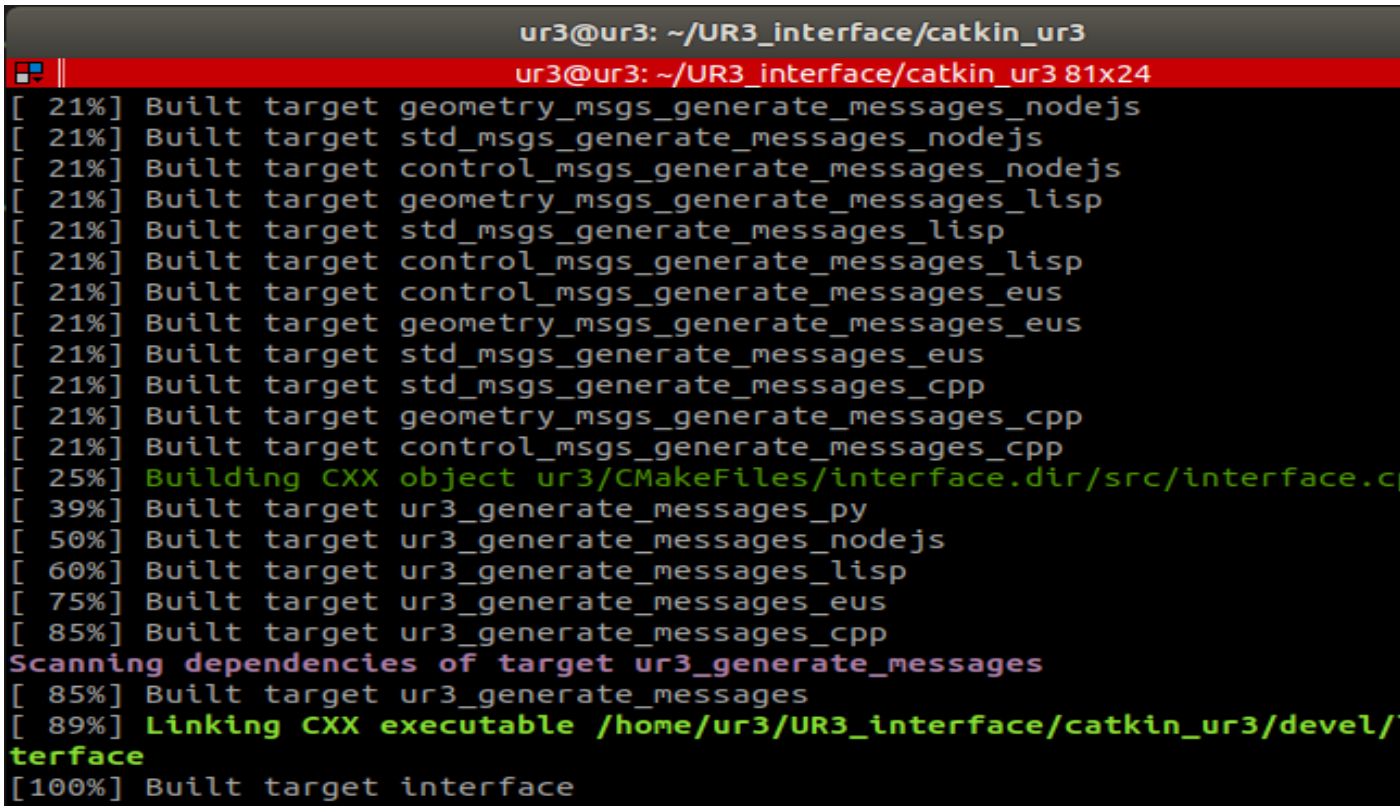


Fig. 11: Figure 11.

The computer setup is ready. In the next step we will learn how to start the ROS (Robot Operating System).

ROS initialization

The demo_1 interface and executables are ready (done in the previous step), now let's start ROS. For that, we will open more sections in Terminator.

Open 4 (four) sections as shown in figure 12 below.

To split the Terminator you can use Ctrl + shift + O to split horizontally or Ctrl + shift + E to split vertically. Remembering that the new sections have to be in the same workspace.

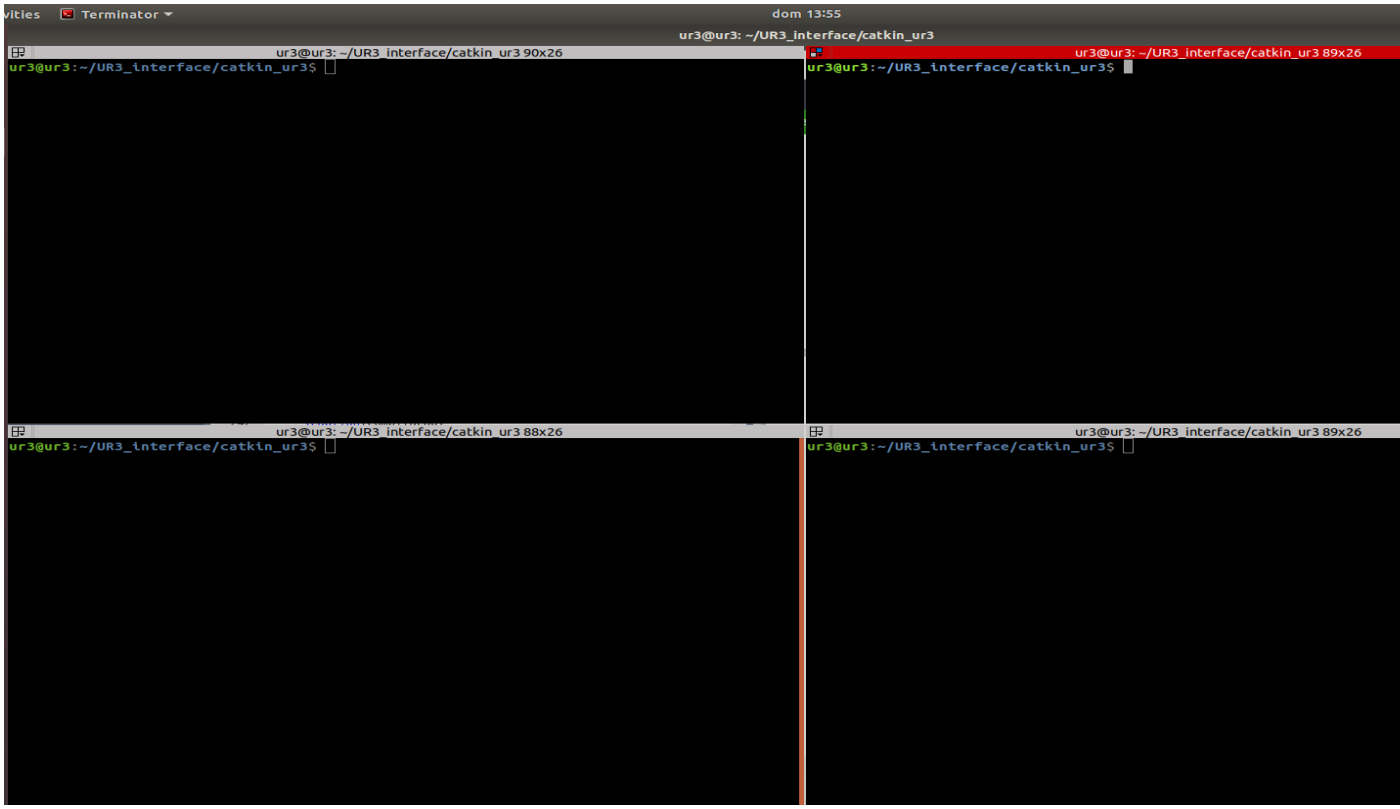


Fig. 12: Figure 12.

Para iniciar o ROS use o comando:

- $ roscore

When ROS is ready, you will receive a message in one of the sections as shown in figure 13 shown below.

NOTE: Don't touch the section that ROS is running.

Running Demo_1

To run **demo_1**, we will first run the **interface** (Use another section of Terminator). For this, we will configure setup of the ROS package with the following command:

- $ source devel/setup.bash

Do this for all sections of the Terminator except for the one running ROS.

Now let's «run» the interface application. To do this, type the command shown below and press ENTER on one of the Terminator sections.

- $ rosrun ur3 interface

If everything is working as expected, the message we will get in return for this command is shown below in figure 14.

When you have confirmation that the robot is ready (UR3 is ready) we can run demo_1.

Now, in another section of Terminator, we will «run» the demo-1 application. To do this, type the command shown below and press ENTER.

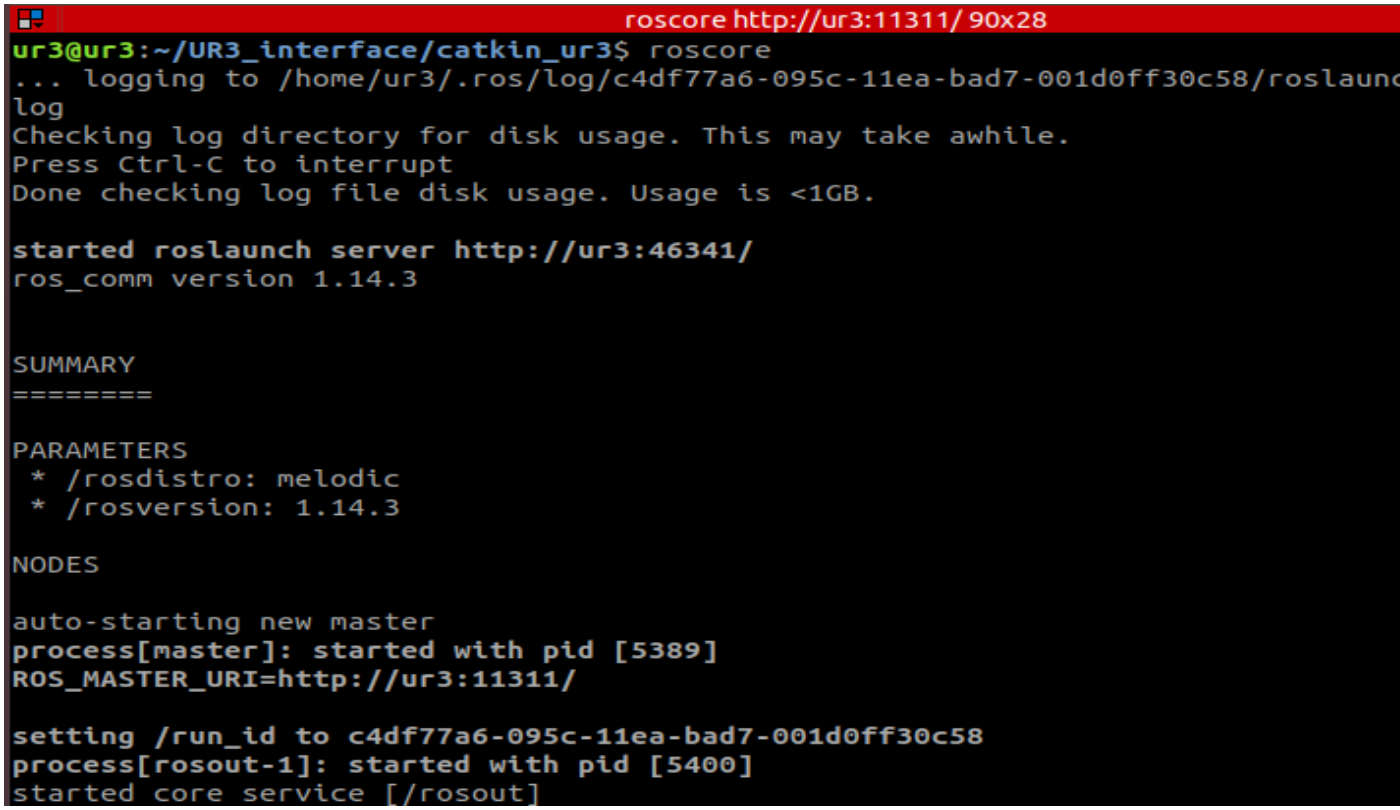


Fig. 13: Figure 13.

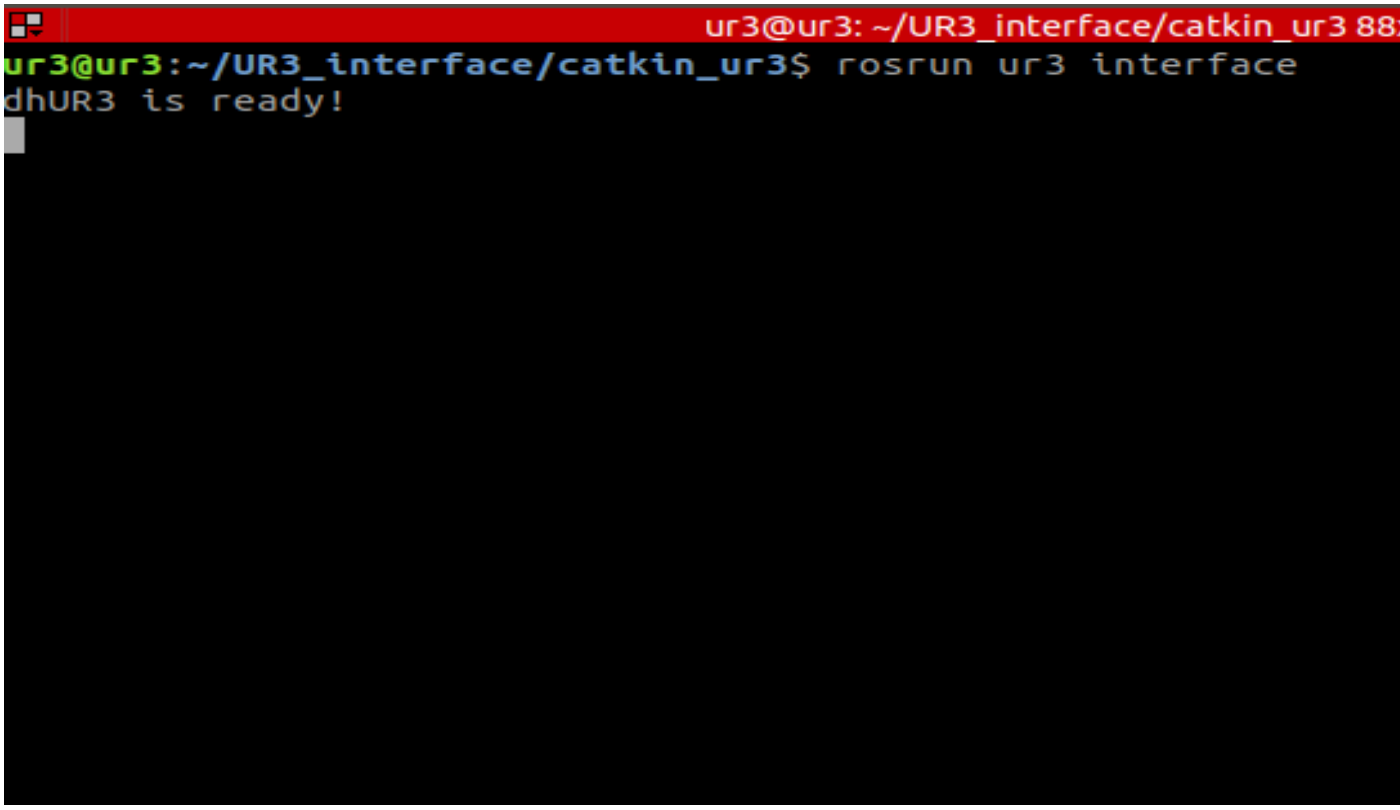


Fig. 14: Figure 14.

- $ rosrun ur3 demo_1

At this point, the robot should start making a repetitive movement and after a few seconds it will stop. After that movement for the robot will have executed demo_1.

View topics

Topics are data structures in which the variables of the robot joints are published. In this robot, there are two topics, but for this experiment, we will focus on just one that will be the topic **arm**.

In order to observe the data present in the topic **arm**, we will type, in another section of **Terminator** the following command:

- $ rostopic echo /arm

Figure 15, shown below, shows the result of this command with its respective variables.

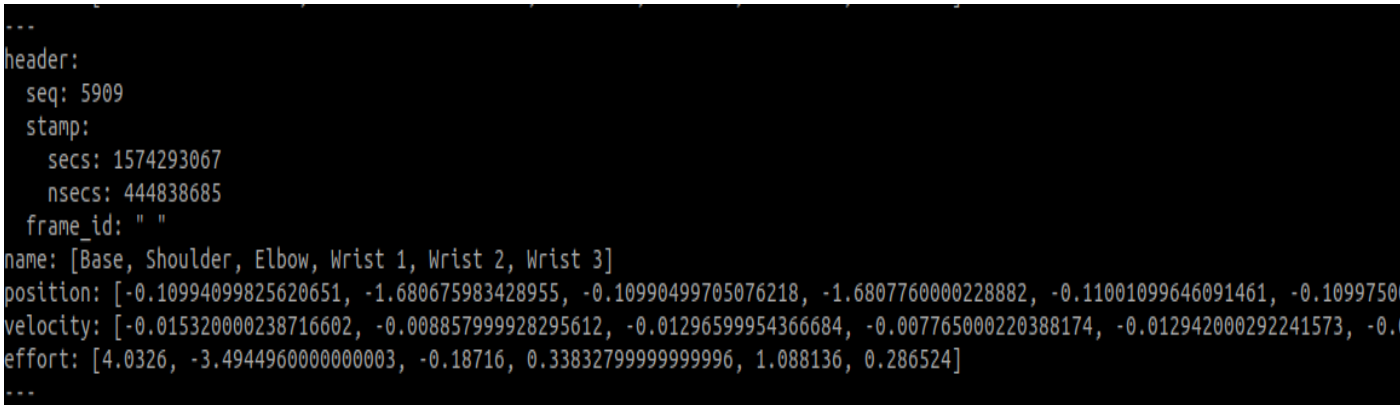


Fig. 15: Figure 15.

You can «kill» this command by typing ctrl + C.

rqt_plot: Graphical data visualization interface

To have a visualization of the joint data in a graphical interface, we will use the application rqt_plot. This application allows the user to observe that of a variable over time.

Given that explanation, let's use the tool. To do this, type the command shown below.

- $ rqt_plot

Figure 16, shown below, shows the application rqt_plot with one of the variables (joint speed 0 (zero)) being shown over time.

To see a particular variable, type/arm/variable[joint number] in Topic below MatPlot.

Saving the experiment to a file

To save the data, just type the following command:

- $ rosbag record /arm

With this command you will write the data for all topics in a bag file.

Turning off Robot and Computer

To turn the robot off, press the **POWER** button, figure 1, and then click **Power Off**.

To shut down the computer, close all **Terminator** tabs and shut down Ubuntu normally.

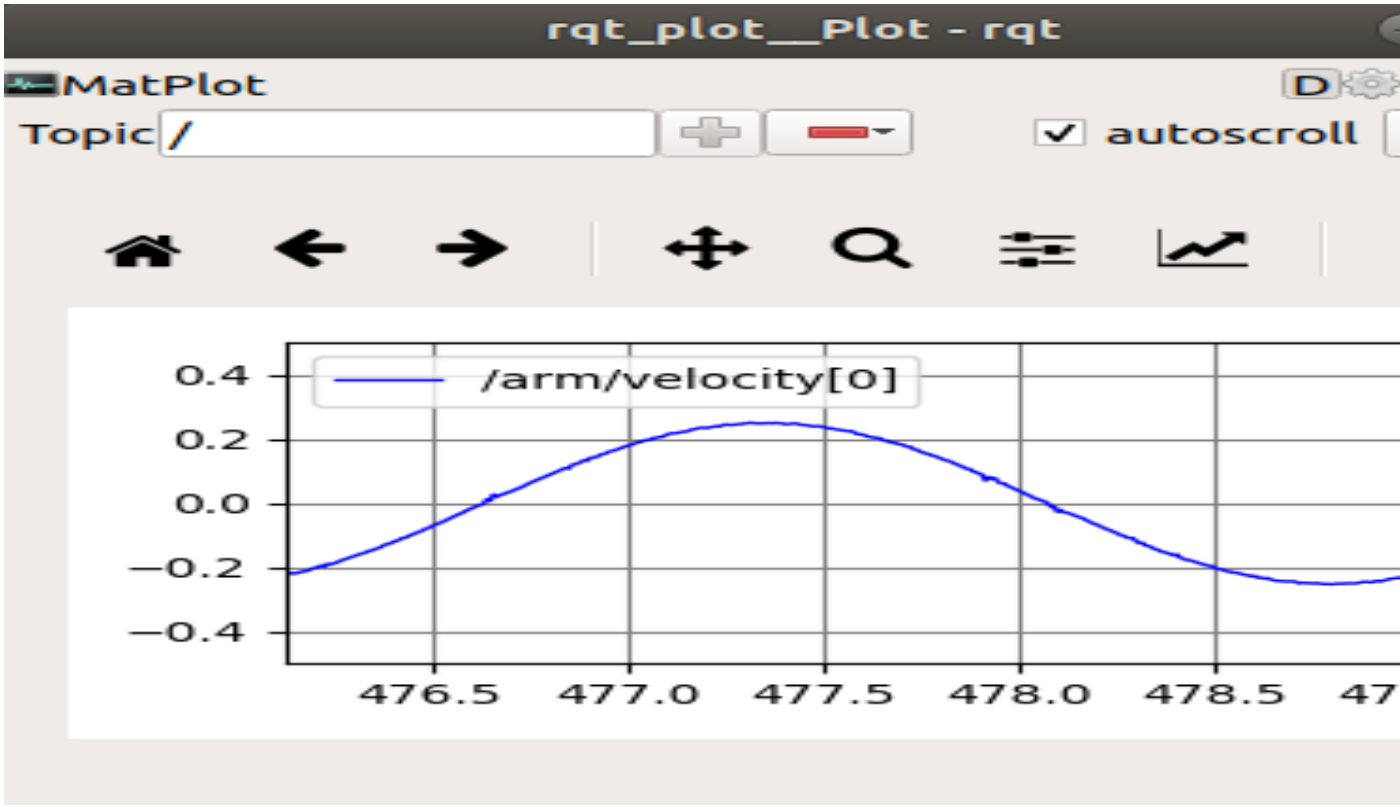
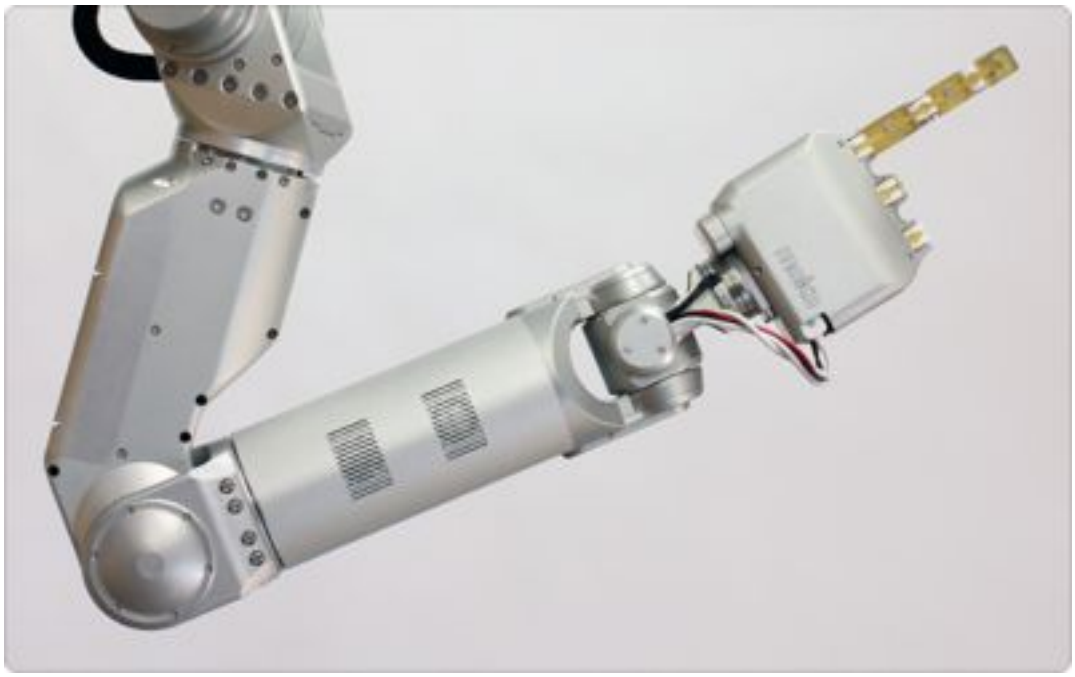


Fig. 16: Figure 16.

3.2 Meka



3.3 Schunk

CAPÍTULO 4

Documentation

Hi,

The official site for Cooperative Robotics is https://cooprobo.readthedocs.io/ or https://cooprobo.rtfd.io/

This site should contain all documentation towards the hardware, software, modelling and tutorial. This site is builded using the official project repo https://github.com/lara-unb/CoopRobo/

4.1 References

About this site

- https://readthedocs.org/
- https://github.com/RobInLabUJI/ROSLab

About ROS programming

- https://github.com/ethz-asl/programming_guidelines/wiki

Best maintainer practices

- https://github.com/leggedrobotics/ros_best_practices/wiki
- https://github.com/leggedrobotics/ros_best_practices/blob/master/ros_package_template/README.md

Programming tips

- https://lara-unb.github.io/dicas-programacao/#/

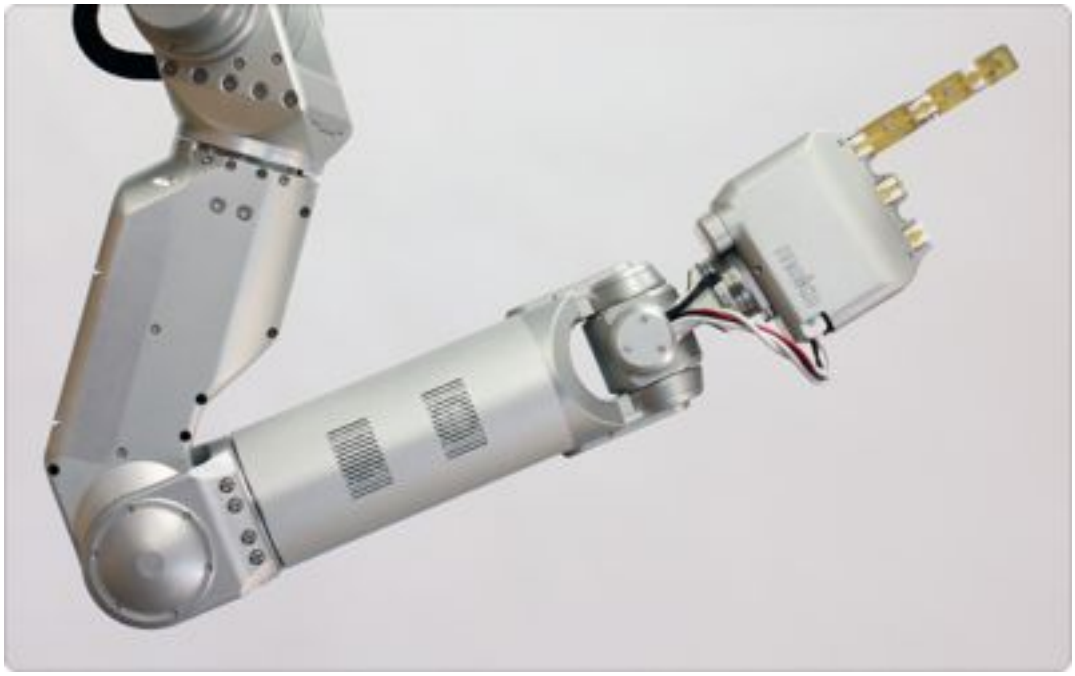
ReadMe Exemple

- https://github.com/Gastd/consensus

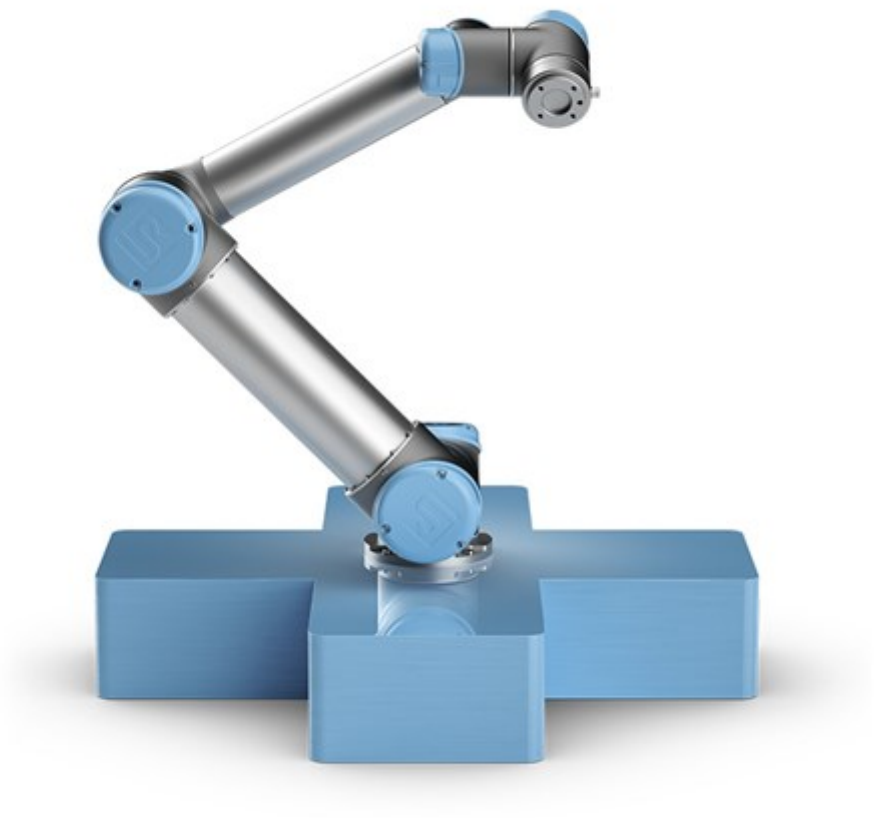
CAPÍTULO 5

Camera

5.1 x



5.2 y



5.2.1 System Description

Overview

The UR3 is a table-top collaborative robot. With its 3 kg payload it is very capable and its small footprint makes it suitable for limited workspace situations. With its infinite turn on the end joint, several activities can be perfomed with grippers attached at robot tool connector.

Some of its applications:

- Laboratory work
- Assembly tasks
- Polishing
- Soldering
- Gluing
- Screwing
- Painting
- Pick and place
- Operating hand tools

- Fume hood tasks

Description

Table 1: UR3 Description

Weight	11.2 kg
Payload	3 kg
Reach	500 mm
Footprint	Ø 128 mm
Degrees of freedom	6 rotating joints
Joint ranges	+/- 360°, infinite rotation on end joint
Speed wrist joints	360 degrees/sec
Other joints	180 degrees/sec.
Noise	Comparatively noiseless
IP classification	IP64

Communication

- TCP/IP 100 Mbit: IEEE 802.3u, 100BASE-TX
- Ethernet socket & Modbus TCP

Control Box

Linux PC

5.2.2 Kinematics

5.2.3 Dynamics

5.2.4 Robots

5.2.5 ROS

5.2.6 Lab 1

5.3 z