Zênite Solar

Equipe bicampeã na categoria livre e pentacampeã em inovação

Construindo uma embarcação inovadora, sustentável, de alta performance, movida a energia solar desde 2013

INSTITUTO FEDERAL
Santa Catarina
Câmpus Florianópolis

DESAFIO SOLAR BRASIL

Implementando uma rede CAN no barco solar

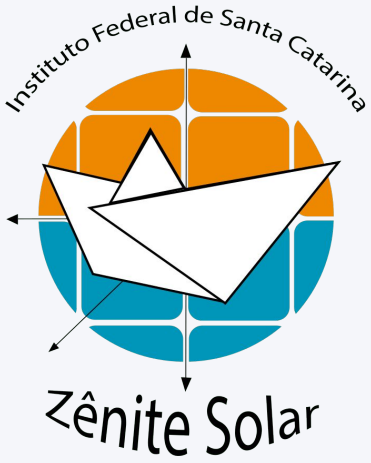
Equipe Zênite Solar
Instituto Federal de Santa Catarina - Campus Florianópolis

Florianópolis, 23 de setembro de 2020.

ZeniteSolar.com | contato@zenitesolar.com

Zênite Solar

- Projeto **Criado** em 2013
- Prêmio de **Inovação Tecnológica** Fernando Amorim
  - 2015, 2016, 2017, 2018 e 2020
- 1° Lugar na Categoria Livre
  - 2015 e 2020
- Mais de **150 estudantes** colaboraram no projeto desde 2013
- **Open source hardware e software** desde 2015



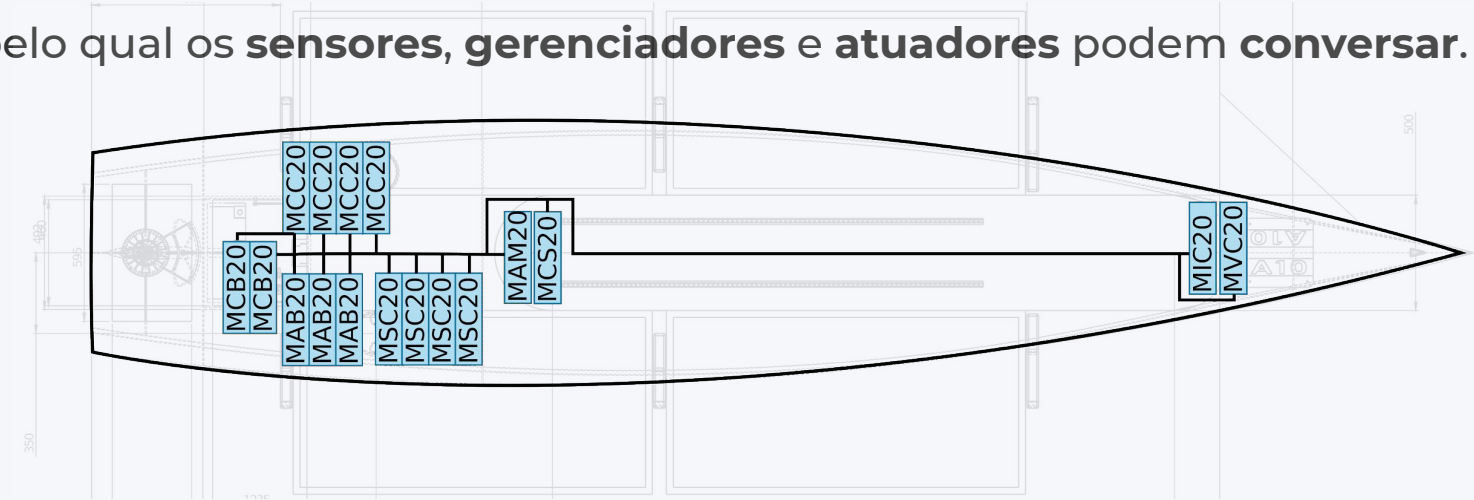
Zênite Solar

Implementando uma rede CAN no barco solar

Sumário

1. A Rede CAN

- CAN (Controller Area Network) é um **protocolo de comunicação** desenvolvido inicialmente para melhorar a comunicação dos *sistemas automotivos*.
- Para nós, a Rede CAN é o **“Sistema nervoso”** do barco - um *caminho único* pelo qual os **sensores**, **gerenciadores** e **atuadores** podem **conversar**.




4.7GB
em 5 dias

2. Por que foi escolhido o CAN?

- Alta confiabilidade e robustez
  - Modos de falha bem conhecidos
  - Diversos mecanismos para detecção de erros
  - Poucos fios
  - Imunidade
- Baixo custo
- Suporta alto fluxo de dados
  - até 1 Mbps
- Flexibilidade
  - Diferentes interconexões, hotplug

3. Como controlar um Motor?

Considerando 3 variáveis: <On/Off>, <Velocidade> e <Sentido>

Solução trivial A:

- <On/Off> : 1 via
- <Sentido> : 1 via
- <Velocidade> : 1 via
- <Referências> : 2 vias
- Total: 5 vias

Solução trivial B:

- <On/Off> : 2 vias trançadas
- <Sentido> : 2 vias trançadas
- <Velocidade> : 3 vias trançadas
- Total: 7 Vias

Desafios: Peso, manutenção, preço, imunidade

Solução ‘wireless’:

- <Alimentação> : 2 vias

Desafios: Manutenção, preço, imunidade, consumo, gestão

Solução com CAN:

- <Alimentação> : 2 vias
- <Comunicação> : 1 a 3 vias
- Total: de 3 a 5 vias

Desafios: Manutenção, gestão


VCC GND SCL SDA
00:17:39.0
BASIC 000
TURBO 100
GND
TX
RX
RST
J302
RST
CLK
MISO
MOSI
J301
U202
U201
TP204
TP203
D305
D304
R320
C201
C203
C204
C205
R201
R202
C202
R317
R316
R315

4. Como fazemos?

4.1. conexões

4.2. sistema modular

4.3. protocolo de aplicação

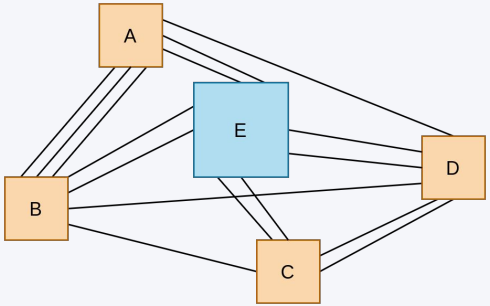
4.4. gestão

4.5. *firmware*

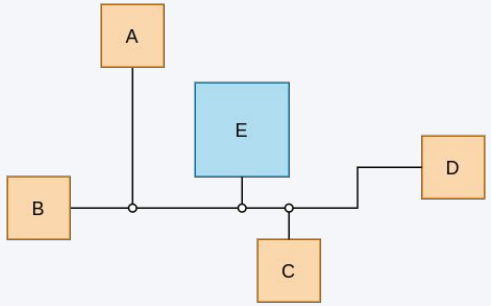
4.1. Como fazemos: conexões

- Cada módulo tem pelo menos 2 portas para conectar vizinhos
- Sistema de conexões ethernet
  - Simples, barato

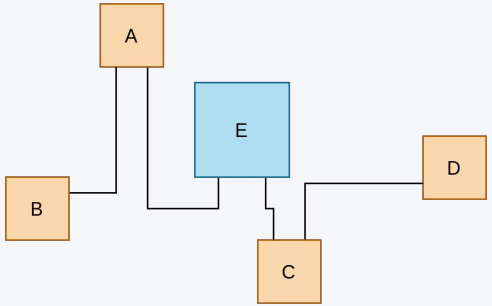
- 5 vias usadas:
  - CAN H
  - CAN L
  - CAN GND
  - +18V (Alimentação dos módulos)
  - GND (Alimentação dos módulos)



Sem CAN



CAN em Star/BUS

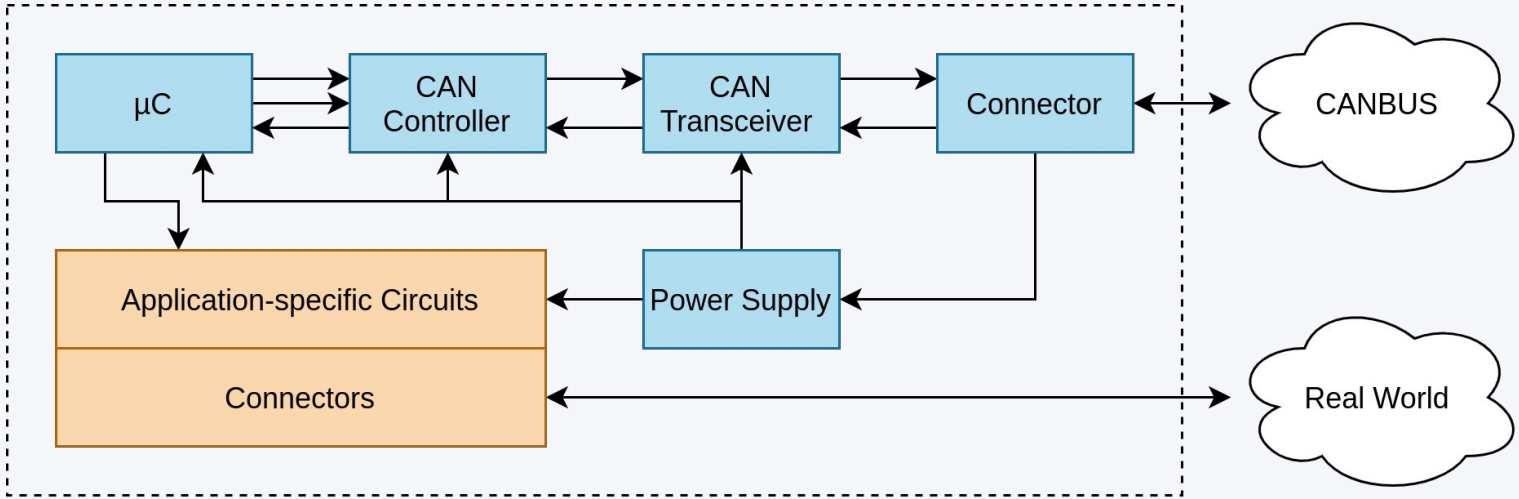


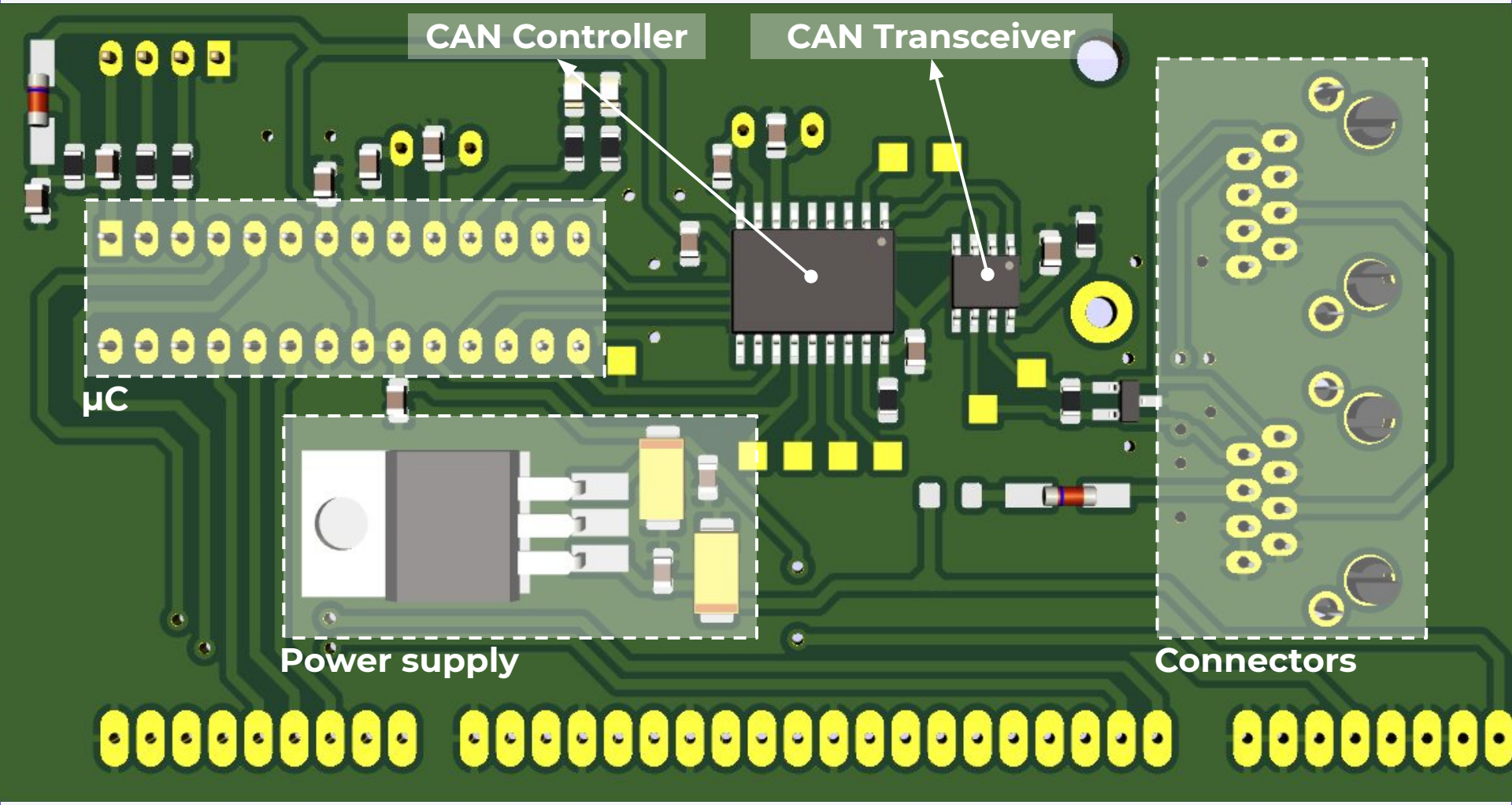
CAN em Daisy chain (Nossa implementação)


II.

4.2. Como fazemos: sistema modular

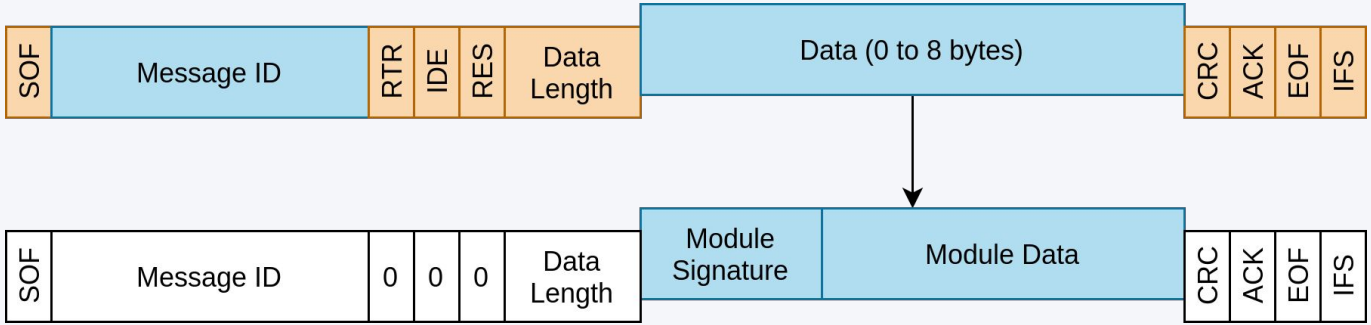
- Todos os módulos compartilham um mesmo design
- Software e Hardware simplificados
- Manutenção facilitada
- Agilidade no desenvolvimento




CAN Controller
CAN Transceiver
µC
Power supply
Connectors

4.3. Como fazemos: protocolo de aplicação

- Existem diversos protocolos de aplicação proprietários
- O protocolo mais utilizado é o CANOpen, que é open-source
- Fizemos um protocolo de aplicação customizado:
  - Mais simples de implementar
  - Mais fácil de manter



4.4. Como fazemos: gestão

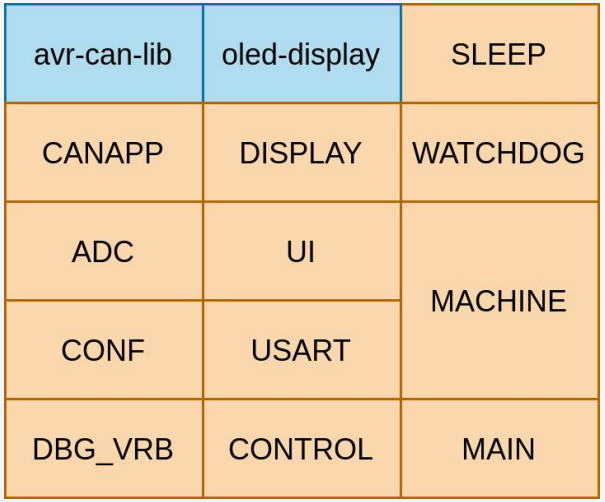
- Todas as mensagens de cada um dos módulos é descrita e mantida num único documento (um script em python)
- Este documento único é mantido num repositório próprio e isolado
  - *github.com/ZeniteSolar/CAN_IDS*
- Uma biblioteca C é gerada a partir desse documento
- Cada módulo usa sempre a última versão da biblioteca
- Cada módulo é testado:
  - individualmente
  - em conjunto com quem se comunica
  - em conjunto com o restante do sistema

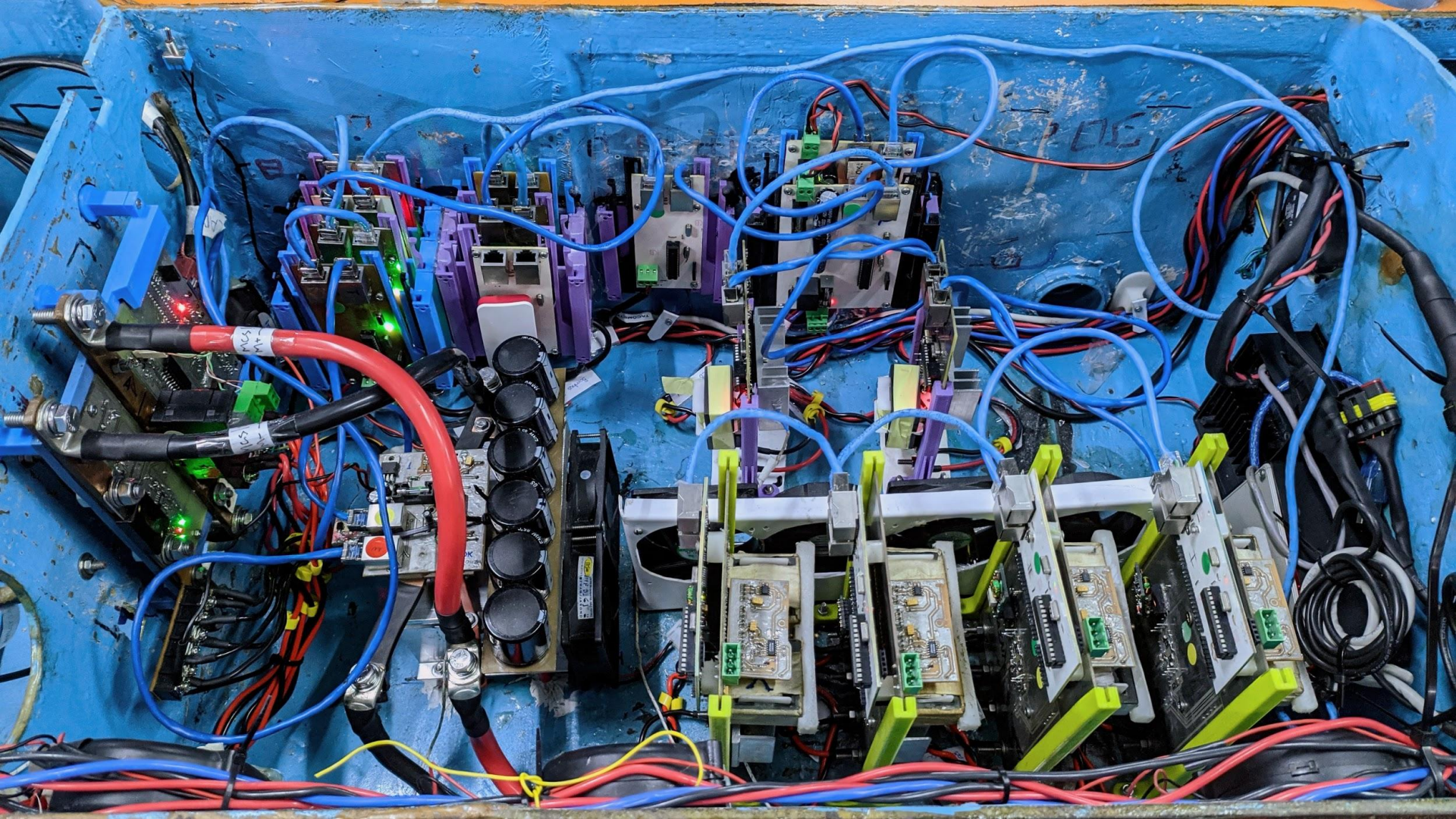
4.5. Como fazemos: firmware

- Bibliotecas externas:
  - “Universelle CAN Blibiothek” (BSD):

    github.com/dergraaf/avr-can-lib
  - “OLED for AVR mikrocontrollers” (GPL-3.0):

    github.com/Sylaina/oled-display
- Código próprio (GPL-3.0):
  - Específico para Atmega328p
  - Separados em diversos módulos reutilizáveis de software
  - Os módulos possivelmente são customizados para cada aplicação





5. Conclusões

- Pontos chave da nossa implementação do CAN:
  - Modularidade com simplicidade
  - Definir padrões:
    - protocolo e sua gestão
    - blocos de software
    - blocos de hardware
    - conexões e parafusos
    - documentação
  - Experimentar e tentar melhorar a cada versão, em pequenos passos
- Recomendações
  - Segurança em 1° Lugar
  - Buscar referências em projetos abertos
  - Começar o quanto antes

6. Referências

- CAN Bus Explained : <https://www.csselectronics.com/screen/page/simple-intro-to-can-bus/language/en>
- CAN vs. RS-485: Why CAN Is on the Move : <https://www.maximintegrated.com/content/dam/files/design/technical-documents/white-papers/can-wp.pdf>
- Comunicação de Tempo Real em uma CAN : <https://www.cin.ufpe.br/~imlm/?Comunica%E7%E3o_de_Tempo_Real_em_uma_CAN>
- Zênite Solar - CAN IDS : <https://github.com/ZeniteSolar/CAN_IDS>
- Universelle CAN Blibiothek : <https://github.com/dergraaf/avr-can-lib>
- OLED for AVR mikrocontrollers : <https://github.com/Sylaina/oled-display>

Obrigado!

/ZeniteSolar

/ZeniteSolar

/Company/ZeniteSolar

/ZeniteSolar

contato@ZeniteSolar.com

ZeniteSolar.com

download dessa apresentação:

ZeniteSolar.com/Downloads/CAN