



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



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
- 1º 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



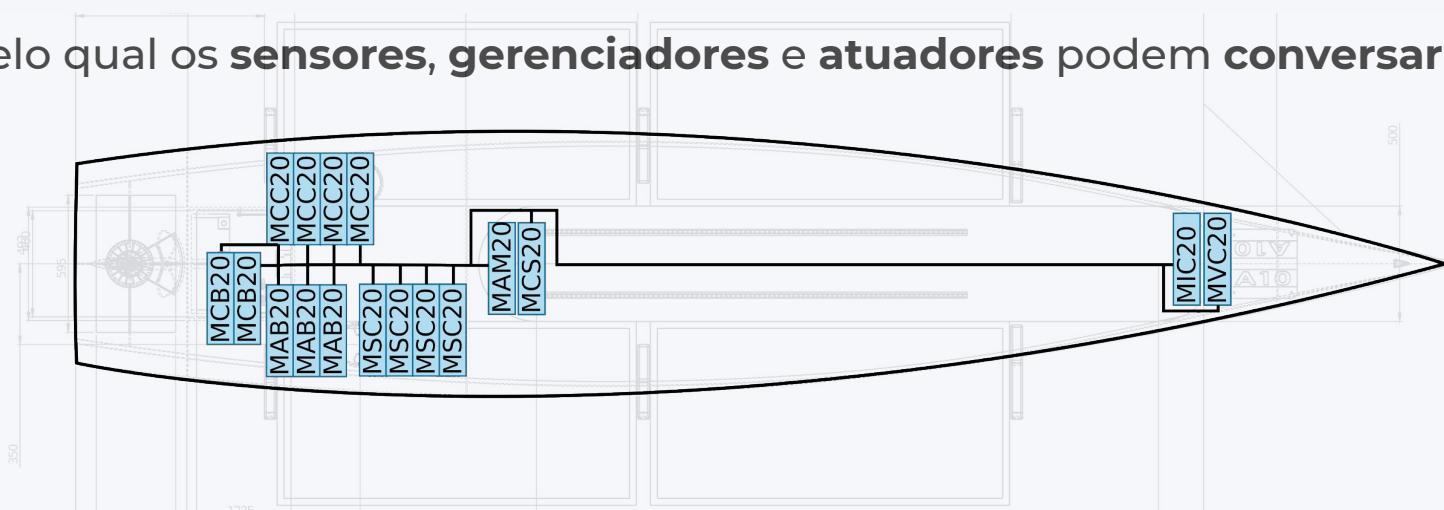
Sumário

1. A Rede CAN
2. Por que foi escolhido o CAN?
3. Como controlar um Motor?
4. Como fazemos
 - 4.1. conexões
 - 4.2. sistema modular
 - 4.3. protocolo de aplicação
 - 4.4. gestão
 - 4.5. firmware
5. Conclusões
6. Referências



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





00:17:39.0

BASIC 000

TURBO 100

R320

R305

R306

R307

C201

C202

C204

C205

U202

U201

TP204

TP203

U201

R201 C205

R202 C202

GND

TX

RX

RST

I302

D407

R404

R405

Q404

C404

L404

C405

R406

Q405

C406

R407

Q406

C407

R408

Q407

C408

R409

Q408

C409

R410

Q409

C410

R411

Q410

C411

R412

Q411

C412

R413

Q414

C415

R416

Q417

C416

R417

Q418

C417

R418

Q419

C418

R419

Q420

C419

R420

Q421

C420

R421

Q422

C421

R422

Q423

C422

R423

Q424

C423

R424

Q425

C424

R425

Q426

C425

R426

Q427

C426

R427

Q428

C427

R428

Q429

C428

R429

Q430

C429

R430

Q431

C430

R431

Q432

C431

R432

Q433

C432

R433

Q434

C433

R434

Q435

C434

R435

Q436

C435

R436

Q437

C436

R437

Q438

C437

R438

Q439

C438

R439

Q440

C439

R440

Q441

C440

R441

Q442

C441

R442

Q443

C442

R443

Q444

C443

R444

Q445

C444

R445

Q446

C445

R446

Q447

C446

R447

Q448

C447

R448

Q449

C448

R449

Q450

C449

R450

Q451

C450

R451

Q452

C451

R452

Q453

C452

R453

Q454

C453

R454

Q455

C454

R455

Q456

C455

R456

Q457

C456

R457

Q458

C457

R458

Q459

C458

R459

Q460

C459

R460

Q461

C460

R461

Q462

C461

R462

Q463

C462

R463

Q464

C463

R464

Q465

C464

R465

Q466

C465

R466

Q467

C466

R467

Q468

C467

R468

Q469

C468

R469

Q470

C469

R470

Q471

C470

R471

Q472

C471

R472

Q473

C472

R473

Q474

C473

R474

Q475

C474

R475

Q476

C475

R476

Q477

C476

R477

Q478

C477

R478

Q479

C478

R479

Q480

C479

R480

Q481

C480

R481

Q482

C481

R482

Q483

C482

R483

Q484

C483

R484

Q485

C484

R485

Q486

C485

R486

Q487

C486

R487

Q488

C487

R488

Q489

C488

R489

Q490

C489

R490

Q491

C490

R491

Q492

C491

R492

Q493

C492

R493

Q494

C493

R494

Q495

C494

R495

Q496

C495

R496

Q497

C496

R497

Q498

C497

R498

Q499

C498

R499

Q500

C499

R500

Q501

C500

R501

Q502

C501

R502

Q503

C502

R503

Q504

C503

R504

</

4. Como fazemos?

4.1. conexões

4.2. sistema modular

4.3. protocolo de aplicação

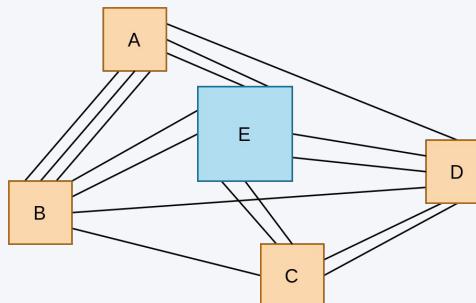
4.4. gestão

4.5. *firmware*

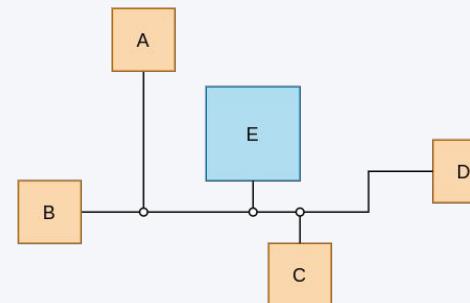


4.1. Como fazemos: conexões

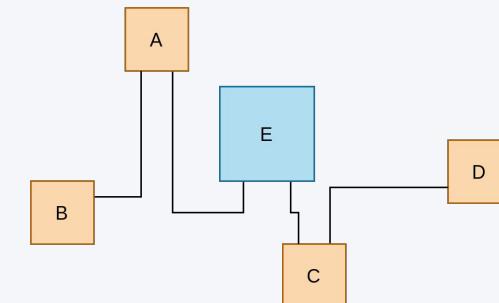
- Cada módulo tem pelo menos 2 portas para conectar vizinhos
- Sistema de conexões ethernet
 - Simples, barato
- 5 vias usadas:
 - CAN H
 - CAN L
 - CAN GND
 - +18V (Alimentação dos módulos)
 - GND (Alimentação dos módulos)



Sem CAN



CAN em
Star/BUS



CAN em Daisy chain
(Nossa implementação)



4.1. Como fazemos: conexões



DB9 e M12



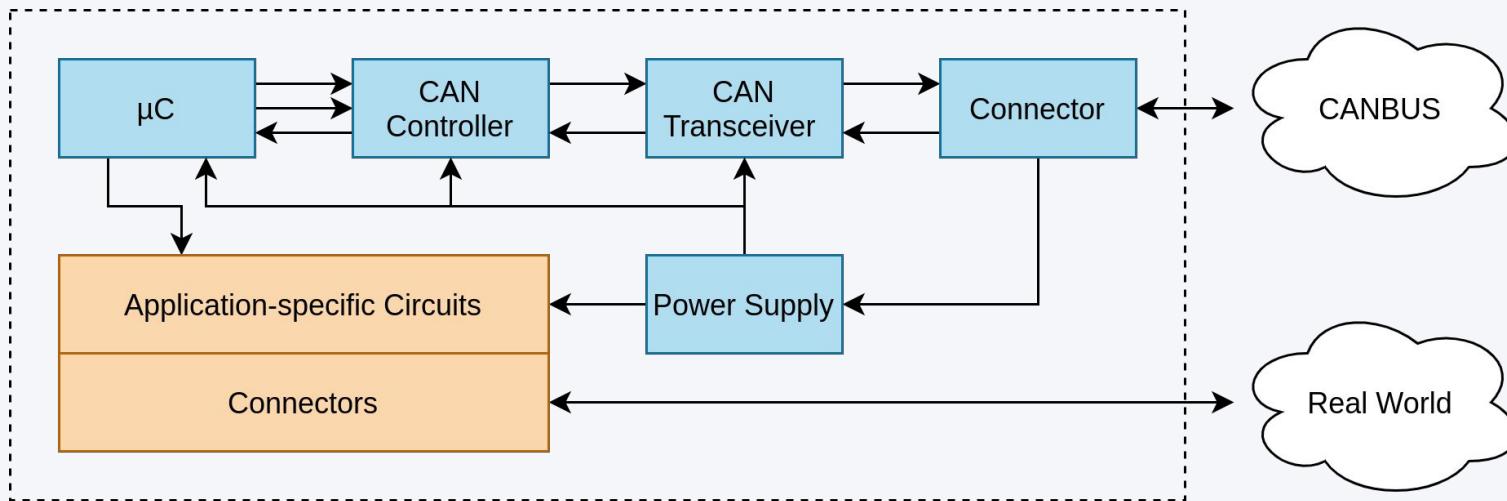
Rj45





4.2. Como fazemos: sistema modular

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



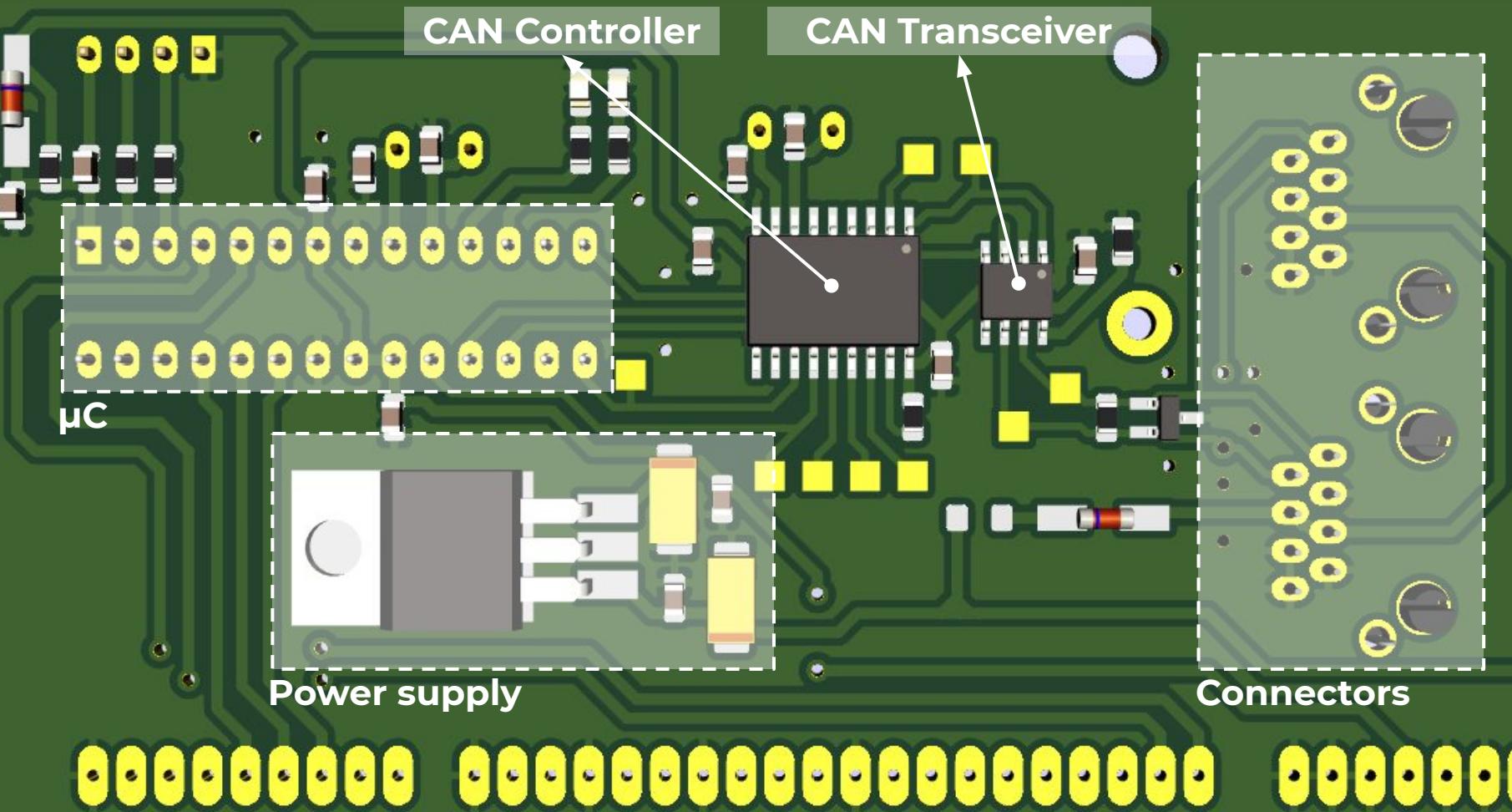
CAN Controller

CAN Transceiver

μ C

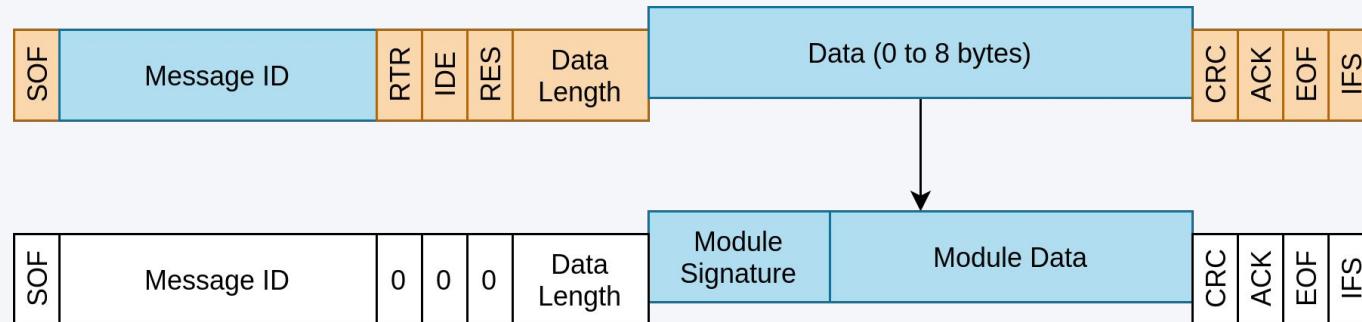
Power supply

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



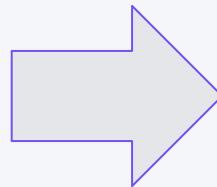
4.4. Como fazemos: gestão

can_ids_generator.py

```

310  ### TOPIC: MEASUREMENTS
311  topic_measurements = can.topic(
312      msg = "measurements",
313      id  = 0b10001,
314      description = "All measurements from the converter"
315  )
316  topic_measurements.describe_byte(
317      name = "output_voltage_l",
318      byte = 1,
319      description = "Average output voltage, byte low",
320      type = "u16",
321      units = "V/100"
322  )
323  topic_measurements.describe_byte(
324      name = "output_voltage_h",
325      byte = 2,
326      description = "Average output voltage, byte high",
327      type = "u16",
328      units = "V/100"
329  )

```



can_ids.h

```

338 // MCB19_1 - MEASUREMENTS - All measurements from the con
339 #define CAN_MSG_MCB19_1_MEASUREMENTS_ID 17
340 #define CAN_MSG_MCB19_1_MEASUREMENTS_LENGTH 8
341 #define CAN_MSG_MCB19_1_MEASUREMENTS_SIGNATURE_BYTE 0 //<
342 #define CAN_MSG_MCB19_1_MEASUREMENTS_SIGNATURE_TYPE "u8"
343 #define CAN_MSG_MCB19_1_MEASUREMENTS_SIGNATURE_UNITS ""
344 #define CAN_MSG_MCB19_1_MEASUREMENTS_OUTPUT_VOLTAGE_L_BYT
345 #define CAN_MSG_MCB19_1_MEASUREMENTS_OUTPUT_VOLTAGE_L_TYP
346 #define CAN_MSG_MCB19_1_MEASUREMENTS_OUTPUT_VOLTAGE_L_UNI
347 #define CAN_MSG_MCB19_1_MEASUREMENTS_OUTPUT_VOLTAGE_H_BYT
348 #define CAN_MSG_MCB19_1_MEASUREMENTS_OUTPUT_VOLTAGE_H_TYP
349 #define CAN_MSG_MCB19_1_MEASUREMENTS_OUTPUT_VOLTAGE_H_UNI

```

can_ids.json

```
{
    "name": "OUTPUT_VOLTAGE_L",
    "description": "Average output voltage, byte low",
    "type": "u16",
    "units": "V/100"
},
```



4.5. Como fazemos: firmware

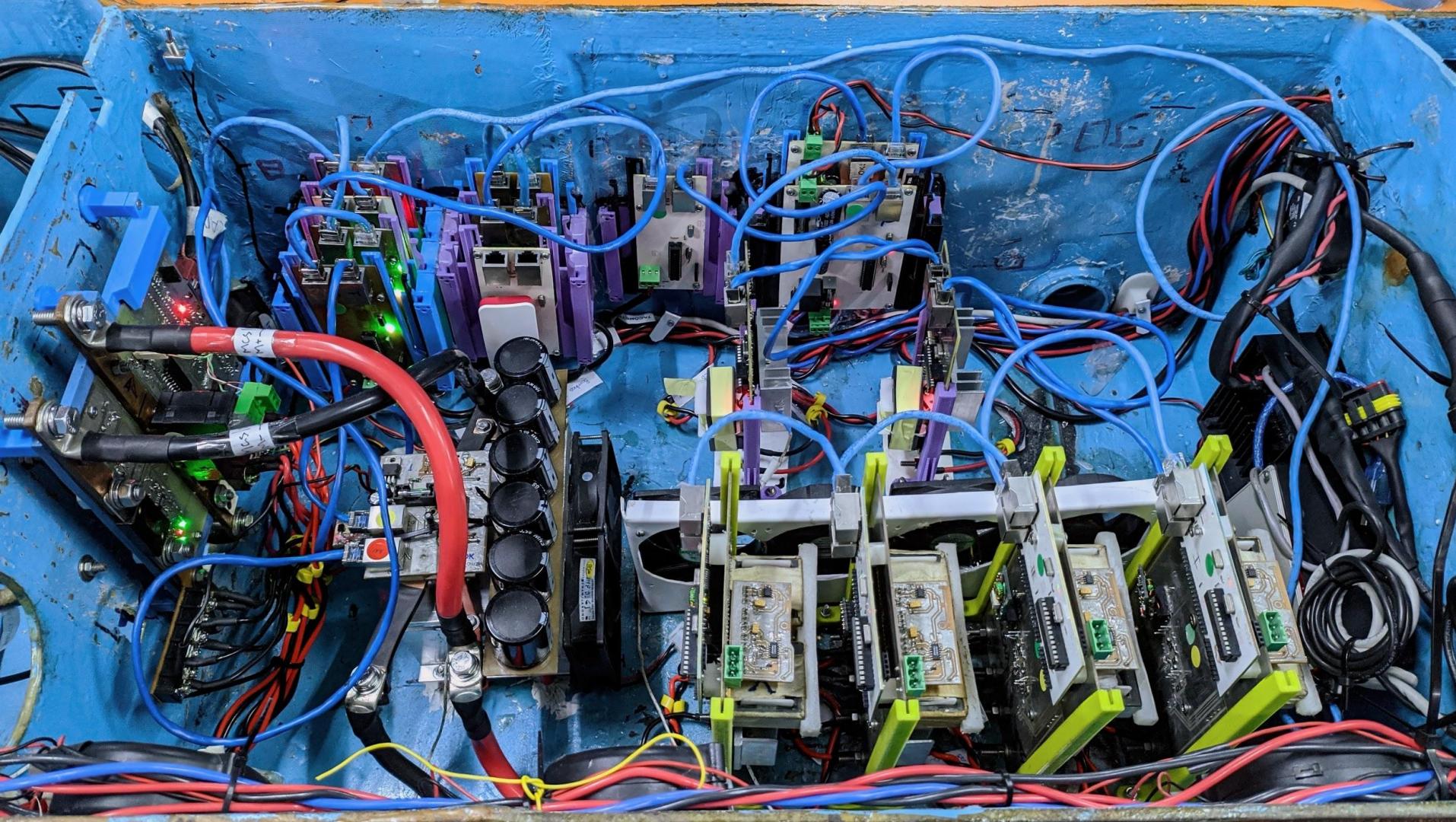
- Bibliotecas externas:
 - “Universelle CAN Blibiothek” (BSD):

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 de software reutilizáveis
 - Os módulos geralmente são customizados para cada aplicação

avr-can-lib	oled-display	SLEEP
CANAPP	DISPLAY	WATCHDOG
ADC	UI	MACHINE
CONF	USART	
DBG_VRB	CONTROL	MAIN





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 1º Lugar
 - Buscar referências em projetos abertos
 - Começar o quanto antes



master

Go to file

Add file

Code



joaoantoniocardoso Update LICENSE

4 days ago 174

.config	Remove trailing spaces	last month
.github	Fix upload dir	last month
control	wip: modelagem do buck	2 months ago
firmware	Fix wrong control flag usage	last month
hardware	Clean unnecessary files	last month
simulations	Add simulation schematic img	2 months ago
.gitignore	gitignore, readme, license	9 months ago
CHANGES.txt	Adding CERN-OHL-S-2.0+ license files	4 days ago
LICENSE	gitignore, readme, license	9 months ago
README.md	Adding CERN-OHL-S-2.0+ license files	4 days ago
cern_ohl_s_v2.pdf	Adding CERN-OHL-S-2.0+ license files	4 days ago
cern_ohl_s_v2.txt	Adding CERN-OHL-S-2.0+ license files	4 days ago
cern_ohl_s_v2...	Adding CERN-OHL-S-2.0+ license files	4 days ago
cern_ohl_s_v2...	Adding CERN-OHL-S-2.0+ license files	4 days ago
gpl.txt	Adding CERN-OHL-S-2.0+ license files	4 days ago
license_badge.svg	Adding CERN-OHL-S-2.0+ license files	4 days ago

About

Modulo de carregamento das baterias auxiliares

zenitesolar.github.io/...

almeta328p buck
battery-charger
solar-boat canbus

Readme

View license

Contributors 5



Languages



CONTROL

Implements the controller for the power converter.

Summary

Members	Descriptions
public inline void control (void)	
public void control_init (void)	
public inline void control_feedback (void)	
public inline float pivo (float r,float y)	
public inline float pii0 (float r,float y)	
union control_flags	

Members

```
PUBLIC INLINE VOID CONTROL (VOID)
PUBLIC VOID CONTROL_INIT (VOID)
PUBLIC INLINE VOID CONTROL_FEEDBACK (VOID)
PUBLIC INLINE FLOAT PIVO (FLOAT R,FLOAT Y)
PUBLIC INLINE FLOAT PII0 (FLOAT R,FLOAT Y)
```

Back to top

Copyright © 2013-2020 Zenite Solar. Distributed by a GNU General Public License v3.0.

Edit this page on GitHub.

This site uses Just the Docs, a documentation theme for Jekyll.

Repositório no github

Documentação automatizada (WIP)



6. Referências

- CAN Bus Explained :
[<https://www.csselectronics.com/screen/page/simple-intro-to-can-bus/language/en>](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>](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%EAo_de_Tempo_Real_em_uma_CAN>](https://www.cin.ufpe.br/~imlm/?Comunica%EAo_de_Tempo_Real_em_uma_CAN)
- CAN CiA : [<https://www.can-cia.org/can-knowledge/>](https://www.can-cia.org/can-knowledge/)
- Zênite Solar - CAN IDS : [<https://github.com/ZeniteSolar/CAN_IDS>](https://github.com/ZeniteSolar/CAN_IDS)
- Universelle CAN Blibliothek : [<https://github.com/dergraaf/avr-can-lib>](https://github.com/dergraaf/avr-can-lib)
- OLED for AVR mikrocontrollers : [<https://github.com/Sylaina/oled-display>](https://github.com/Sylaina/oled-display)



Obrigado!



/ZeniteSolar



/ZeniteSolar



/Company/ZeniteSolar



/ZeniteSolar



contato@ZeniteSolar.com



ZeniteSolar.com



download dessa apresentação:

ZeniteSolar.com/downloads/CAN

