Universidade Federal da Bahia - Escola Politécnica - UFBA
Laboratório Integrado II-A
Alunos(a): Ana Clara Batista, André Paiva Conrado Rodrigues
Projeto ZMP

1 - Introdução

Trabalho referente a matéria: Laboratório Integrado II-A do curso Engenharia da Computação da Escola Politécnica da UFBA realizado no primeiro semestre de 2021. Nossa parte do trabalho consiste no projeto de cálculo de referência de forças para rodas de um carro a partir de leitura de IMU de 9 GdL e feedback de RPM/Força das rodas.

A ideia é de implementação em Intel Galileo, sendo os esquemáticos e código feitos pensando numa futura implementação nessa plataforma.

2 - Diagramas

Primeiramente foi realizado um diagrama de blocos do projeto especificando cada funcionalidade e cálculos do mesmo, mostrado na Figura 1. Recolhemos informações do sensor IMU (magnetômetro, momento angular e aceleração) e com ele realizamos o cálculo do centro de massa. Após esse cálculo, também realizamos o cálculo das referências de forças e enviamos esses dados para os módulos das rodas dianteiras e traseiras através do módulo CAN. Do módulo CAN também recebemos informações das rodas.

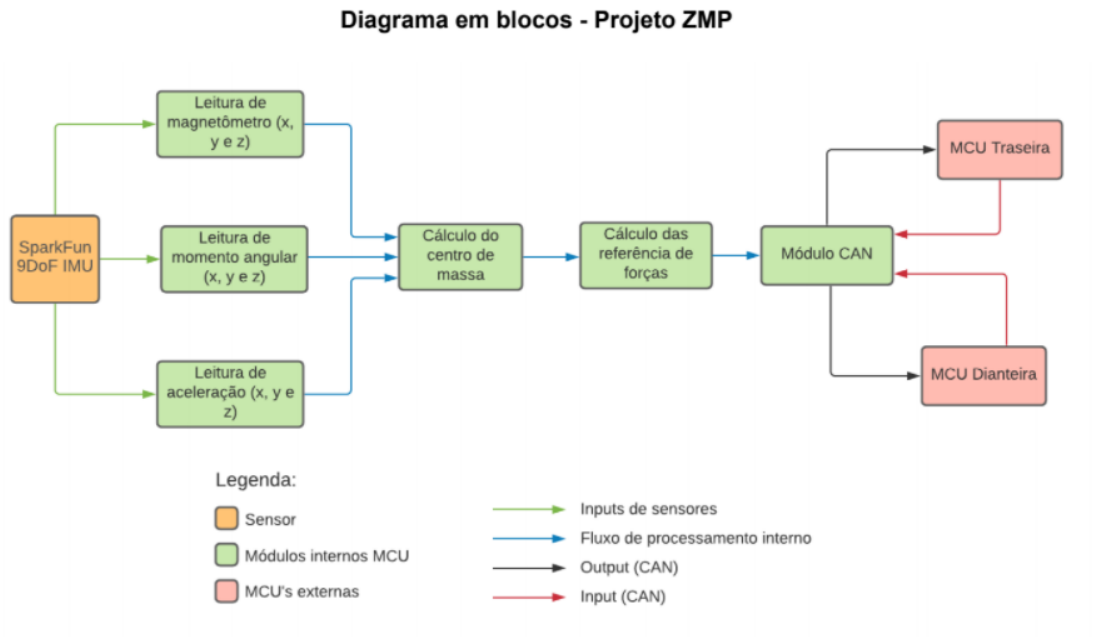

Figura 1 - Diagrama de blocos

Na Figura 2 consta o diagrama esquemático, onde mostra todas as ligações entre a placa Intel Galileo de geração 2 com o sensor e o módulo CAN. Essas mesmas ligações podem também ser visualizadas de forma mais colorida através da Figura 3 onde consta o diagrama de montagem.

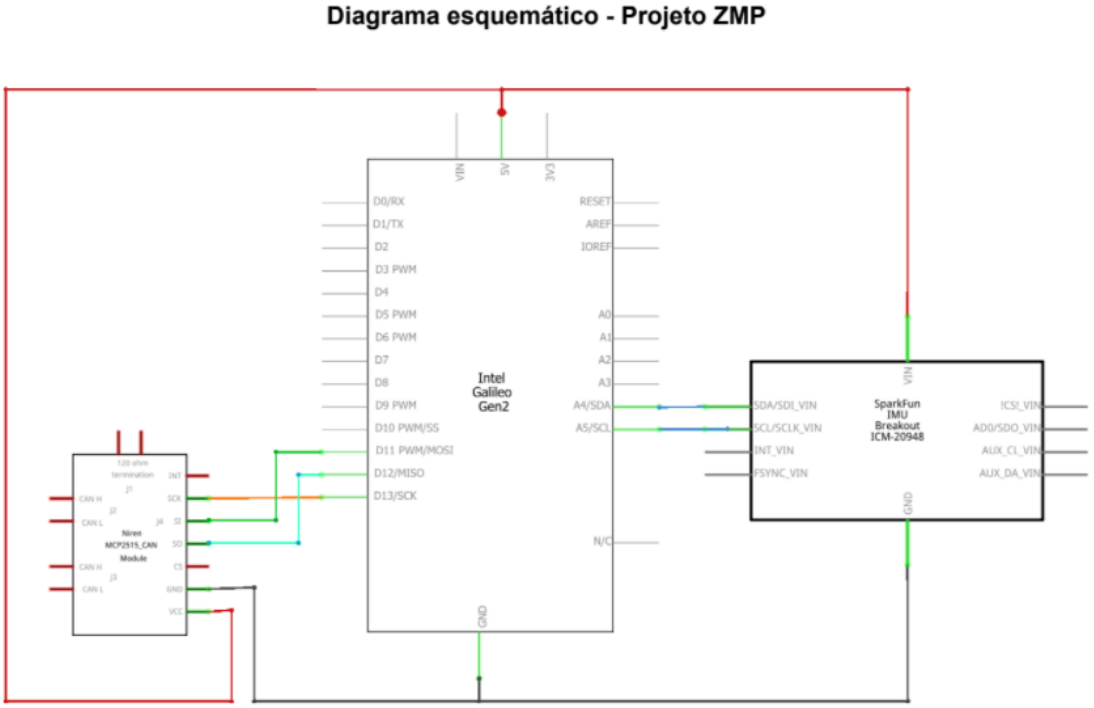

Figura 2 - Diagrama esquemático

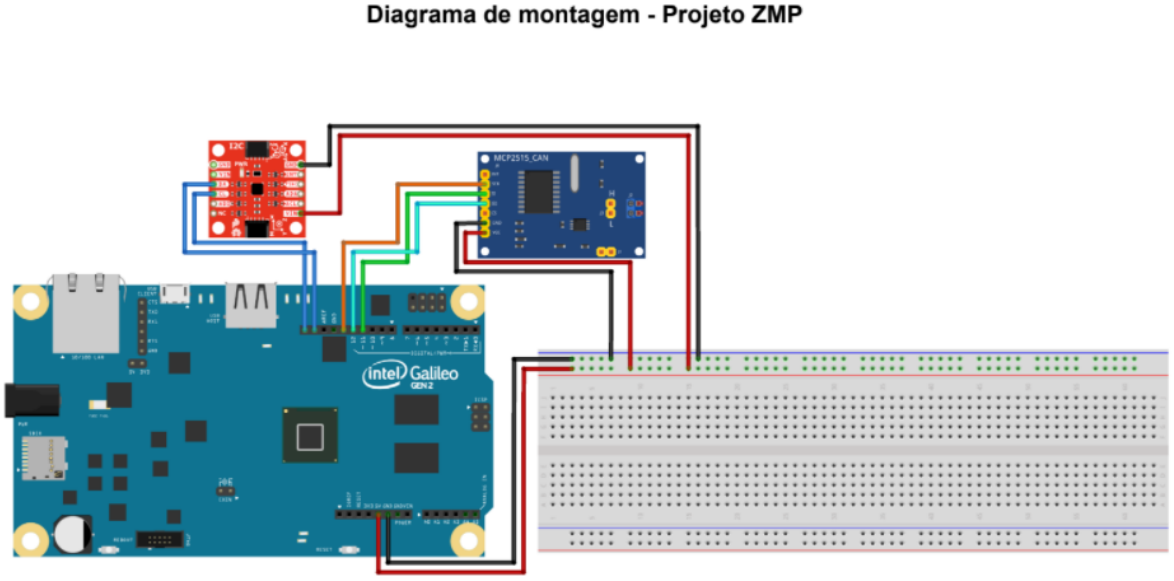

Figura 3 - Diagrama de montagem

3 - Código e implementação

Até o presente momento, foi criado um código *.ino* com as funções de escrita/leitura em barramento CAN e leitura da IMU. A leitura das mensagens foi feita considerando que a MCU das rodas dianteiras está emitindo mensagem com ID 0x02 e a MCU das rodas traseiras está emitindo mensagem com ID 0x03 como consta na Figura 4.

```
//Definição de constantes (ID's de mensagens CAN), Considerando
//0x02 ID da mensagem da MCU das rodas dianteiras
//0x03 ID da mensagem da MCU das rodas traseiras
//0x04 ID da mensagem da MCU de cálculo da ZMP (esta)
#define MCU_DIANT 0x02
#define MCU_TRAS 0x03
#define MCU_ZMP_TRAS 0x04
#define MCU_ZMP_DIANT 0x05
```

Figura 4 - Deginição de constantes

A mensagem que escreveremos contendo os dados de referência das rodas traseiras será enviada com ID 0x04, e a mensagem que escreveremos contendo os dados de referência das rodas dianteiras será enviada com ID 0x05. Cada mensagem lida/escrita terá 8 bytes de tamanho.

Tanto para a leitura das rodas dianteiras quanto para a leitura das rodas traseiras, foi considerado o byte de índice maior como byte mais significativo e considerar bit de maior magnitude neste byte como bit de sinal como pode ser visto na Figura 5. A organização de dados do pacote recebido pode ser exemplificado da seguinte maneira:

```
//Função de leitura do módulo CAN
bool leituraCAN(int id_busca){
  bool exito = false;
  if(mcp2515.readMessage(&leitura) == MCP2515::ERROR_OK){
    if(leitura.can_id == MCU_DIANT && id_busca == MCU_DIANT){
      if(exito == false) exito = true;
      for(int i = 0; i < leitura.can_dlc; i++) read_dianteira[i] = leitura.data[i];
    }else if(leitura.can_id == MCU_TRAS && id_busca == MCU_TRAS){
      if(exito == false) exito = true;
      for(int i = 0; i < leitura.can_dlc; i++) read_traseira[i] = leitura.data[i];
    }
  }
  return exito;
}
```

Figura 5 - Função de Leitura

- leitura.data[1] e leitura.data[0]: Leitura de força da roda direita
- leitura.data[3] e leitura.data[2]: Leitura de força da roda esquerda
- leitura.data[5] e leitura.data[4]: Leitura de ângulo da roda direita (dianteira)
- leitura.data[7] e leitura.data[6]: Leitura de ângulo da roda esquerda (dianteira)

Para escrita de dados, a organização de dados do pacote recebido também foi considerado o byte de índice maior como byte mais significativo e considerar bit de maior magnitude neste byte como bit de sinal como visto na Figura 6, sendo montado da seguinte forma:

- escrita.data[1] e escrita.data[0]: Escrita da referência de força da roda direita
- escrita.data[3] e escrita.data[2]: Escrita da referência de força da roda esquerda
- escrita.data[5] e escrita.data[4]: Escrita da ângulo da roda direita (dianteira)
- escrita.data[7] e escrita.data[6]: Escrita da ângulo da roda esquerda (dianteira)

```
//Função de escrita pelo módulo CAN
void escritaCAN(){
  escrita.can_id = MCU_ZMP_DIANT;
  for(int i = 0; i < escrita.can_dlc; i++) escrita.data[i] = write_dianteira[i];
  mcp2515.sendMessage(&escrita);
  escrita.can_id = MCU_ZMP_TRAS;
  for(int i = 0; i < escrita.can_dlc; i++) escrita.data[i] = write_traseira[i];
  mcp2515.sendMessage(&escrita);
}
```

Figura 6 - Função de Escrita

Para a função de cálculo de referência, visto na Figura 7, realizamos um cálculo arbitrário que não condiz com rotinas reais de cálculo de referência de torque de rodas por falta de conhecimento de sistemas de controle, sendo importante, portanto, desconsiderar qualquer dado numérico. Para isso foi realizada uma transformação de dados read_dianteira[] e read_traseira[] em int e o uso desses dados convertidos e dados da leitura da IMU (acc[], gyr[] e magne[]) para fazer o cálculo de referência de torque das rodas. Por fim, foi feita a conversão de cálculo de cada roda e cálculo dos ângulos (rodas dianteiras) em sequência de 2 bytes cada e gravação em write_dianteira[] e write_traseira[] na sequência estabelecida para escrita.data[].

```
//Função de cálculo de força de referência das rodas
void calculoRef(){ //CÁLCULO ARBITRÁRIO, FAVOR DESCONSIDERAR SIGNIFICADO
  unsigned int ref[3], refDiantEsq, refDiantDir, refTrasEsq, refTrasDir, refAnguloEsq, refAngulo
  for(int i = 0; i < 3; i++)
    ref[i] = abs(acc[i]*gyr[i]/magne[i]); //DESCONSIDERE QUALQUER COISA DEPOIS DESSA LINHA
  refDiantEsq = (read_dianteira[3]*256) + read_dianteira[2])/(5*ref[0]);
  refDiantDir = (read_dianteira[1]*256) + read_dianteira[0])/(5*ref[1]);
  refTrasEsq = (read_traseira[3]*256) + read_traseira[2])/(5*ref[0]);
  refTrasDir = (read_traseira[1]*256) + read_traseira[0])/(5*ref[1]);
  refAnguloEsq = (read_dianteira[7]*256) + read_dianteira[6])/ref[2];
  refAnguloDir = (read_dianteira[5]*256) + read_dianteira[4])/ref[2];
  write_dianteira[0] = refDiantDir % 256;
  write_dianteira[1] = refDiantDir / 256;
  write_dianteira[2] = refDiantEsq % 256;
  write_dianteira[3] = refDiantEsq / 256;
  write_dianteira[4] = refAnguloDir % 256;
  write_dianteira[5] = refAnguloDir / 256;
  write_dianteira[6] = refAnguloEsq % 256;
  write_dianteira[7] = refAnguloEsq / 256;
  write_traseira[0] = refTrasDir % 256;
  write_traseira[1] = refTrasDir / 256;
  write_traseira[2] = refTrasEsq % 256;
  write_traseira[3] = refTrasEsq / 256;
  write_traseira[4] = 0;
  write_traseira[5] = 0;
  write_traseira[6] = 0;
  write_traseira[7] = 0;
}
```

Figura 7 - Função de Cálculo

4 - Conclusão

Por fim, utilizamos conhecimentos de programação de Arduino para tal projeto, considerando que a placa Galileo é compatível com tal arquitetura. Conseguimos aprender e implementar noções de microprocessadores de forma útil e prática dentro da disciplina com as limitações que um semestre EAD nos proporcionou. Temos como produto final um projeto esquematizado e um pré-código bem encaminhado.

5 - Referências e Links

- [https://github.com/xornotor/ProjetoZMP] Link do projeto no Github
- [https://www.intel.com.br/content/www/br/pt/support/articles/000005912/boards-and-kits/intel-galileo-boards.html] Documentação do Intel Galileo, acessada em Maio/2021
- [https://fritzing.org/download/] Programa utilizado para realização dos diagramas esquemático e de montagem
- [https://www.sparkfun.com/products/15335?_ga=2.254090894.705802866.1623290115-1052118588.1622159986] Módulo de sensores imaginado para utilização, acessado em Maio/2021
- [https://github.com/sparkfun/SparkFun_Qwiic_9DoF_IMU_Breakout] Documentação do módulo de sensores, acessada em Maio/2021
- [https://github.com/sparkfun/SparkFun_ICM-20948_ArduinoLibrary] Biblioteca arduíno para uso do módulo de sensores, acessado em Maio/2021
- [https://github.com/autowp/arduino-mcp2515] Biblioteca arduíno para do módulo CAN, acessado em Maio/2021