UNIVERSIDADE DE SÃO PAULO

ESCOLA DE ENGENHARIA DE SÃO CARLOS

DEPARTAMENTO DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

SEL318 – Laboratório de Circuitos Eletrônicos III

Sensor do Volume de Combustível do Carro Baja

Autor: Bruno Caceres Carrilho 7656508

São Carlos

2017

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

CAPÍTULO 1

1. Introdução

O projeto Baja SAE consiste na construção de protótipos de carros para competições off-road por universitários, envolvendo várias etapas de projeto. A Equipe EESC USP Baja SAE possui subsistemas dedicados à partes especificas do projeto do carro, e o objetivo deste trabalho é a construção de um sensor para a captação do volume de combustível do tanque do carro para o subsistema da eletrônica.

Para solucionar este problema cogitou-se diversas soluções, como utilização de boias, entretanto, por restrição de segurança não é permitido que fios elétricos saiam do tanque de combustível. Portanto, pensou-se em outras duas soluções, uma utilizando fibras ópticas, que entrariam pela tampa percorreriam toda a extensão do tanque e retornariam para a tampa, assim fazendo-se possível a detecção do volume pela variação da quantidade de luz detectada, uma vez que a variação do volume de combustível causa uma variação no índice de refração do meio externo à fibra, assim variando a quantidade de luz dispersada pela fibra. Entretanto, como não possuímos o detector de luz para fibra ópticas essa ideia foi posta de lado.

A outra solução, a qual foi realizada neste projeto, funciona pelo princípio da variação de capacitância pela variação do dielétrico entre as placas do capacitor, ou seja, foi proposto que o combustível seja o dielétrico do capacitor. Mas como dito anteriormente não é permitido que fios elétricos saiam do tanque, porém aproveitando detalhes construtivos do tanque de combustível pode-se empregar esse capacitor externo ao tanque de tal modo que o combustível seja o dielétrico.

CAPÍTULO 2

2. Embasamento teórico

2.1. Sensor de Nível Capacitivo

Levando-se em consideração um capacitor de placas paralelas podemos equacionar a variação de capacitância em relação a variação do dielétrico do capacitor.

$$C = \frac{\varepsilon A}{d} \tag{1}$$

Expandindo a equação do capacitor em dois capacitores de áreas variáveis, um para o dielétrico do ar, e o outro para o dielétrico do liquido, teremos:

$$C = \frac{\varepsilon_o w(L-h)}{d} + \frac{\varepsilon_l wh}{d} \tag{2}$$

$$C = \frac{w(\varepsilon_o(L-h) + \varepsilon_l h)}{d} \tag{3}$$

$$C = \frac{w(\varepsilon_o(L-h) + \varepsilon_l h)}{d} \tag{4}$$

$$C = \frac{(\varepsilon_l - \varepsilon_o)wh}{d} + \frac{\varepsilon_o wL}{d} \tag{5}$$

Observando a equação (5) nota-se um comportamento linear da variação do liquido com a variação da capacitância.

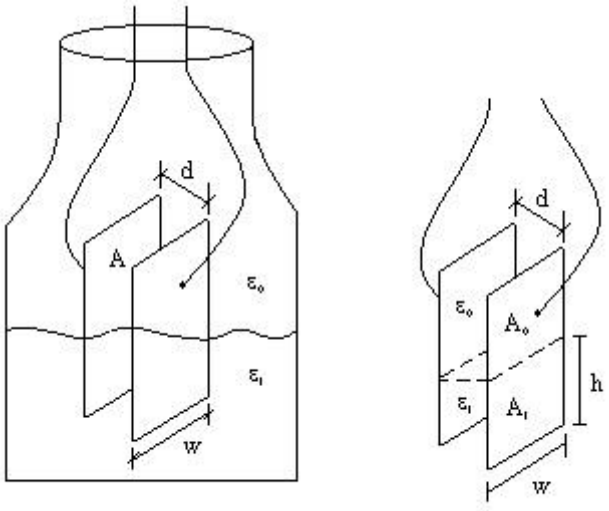

Figura 1: Sensor de Nível Capacitivo.

Com isso pode-se montar circuitos para estimar a capacitância, e portanto, o nível de liquido.

2.2. Circuito Integrado 555

O circuito integrado 555 possui diversas configurações, entre elas duas importantes para este projeto, são elas: configuração astável e configuração monoestável.

2.2.1. Configuração Astável

A configuração astável produz uma oscilação (no pino 3) de saída do CI. Pode-se observar essa configuração na Figura 2.

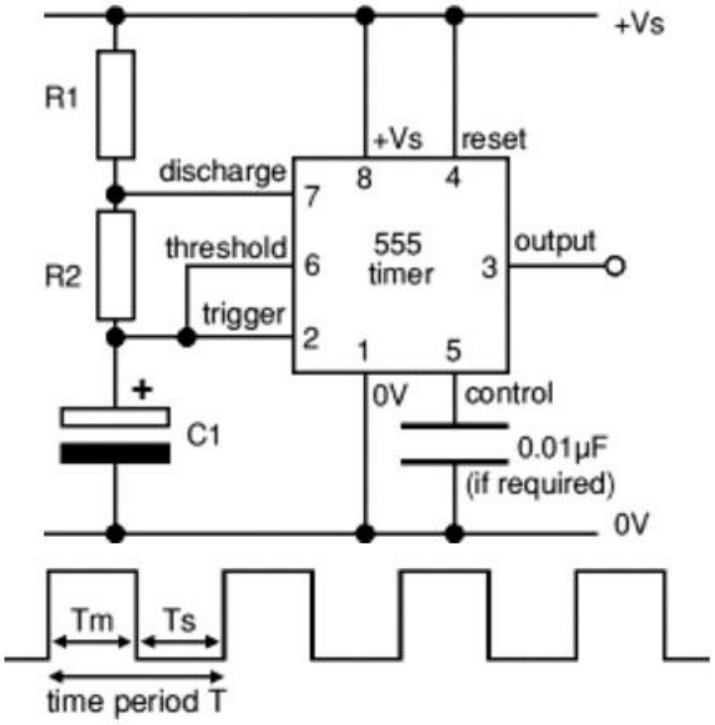

Figura 2: Configuração astável do CI 555.

No circuito astável pode-se alterar o tempo Tm, e Ts. Quando o capacitor carrega até $\frac{1}{3}$Vs, o trigger do 555 dispara e a tensão no pino de saída para tensão Vs, e quando a tensão no capacitor chega a $\frac{2}{3}$Vs, o 555 corta a saída e o pino de saída para Vd, que no caso é o terra (0V).

Pode-se calcular o duty cycle, período e frequência pelas seguintes equações:

$$T_m = 0{,}7(R_1 + R_2)C_1 \quad (6)$$

$$T_s = 0{,}7R_2C_1 \quad (7)$$

$$T = 0{,}7(R_1 + 2R_2)C_1 \quad (8)$$

$$f = \frac{1{,}4}{(R_1 + 2R_2)C_1} \quad (9)$$

2.2.2. Configuração Monoestável

A configuração monoestável do CI 555 é a configuração que apenas o estado logico baixo ‘0’ é estável, com isso pode-se criar um circuito que permaneça no estado logico alto ‘1’ por tempo predeterminado, utilizando-se um capacitor na configuração apresentada pela Figura 3.

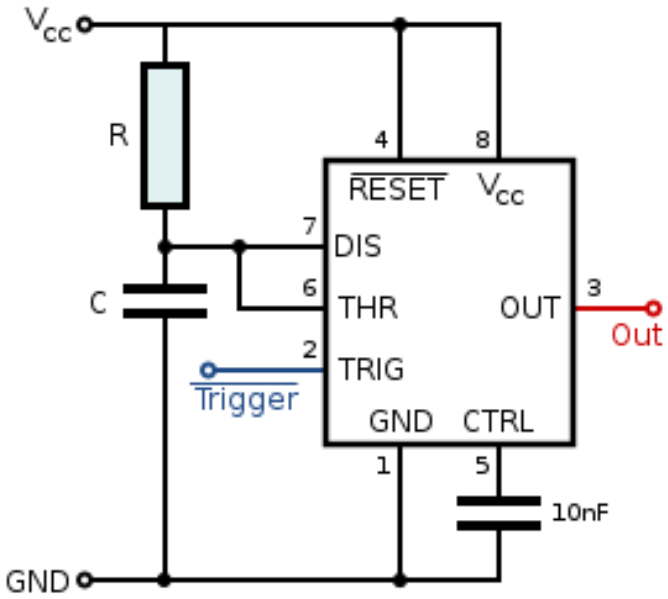

Figura 3: Configuração monoestável do CI 555.

No circuito monoestável quando o trigger é disparado (nível lógico baixo ‘0’) a saída do CI (pino 3) vai para nível logico alto, e permanece em nível logico alto até que a tensão no capacitor atinja $\frac{2}{3}$Vs, então a saída do circuito ira para nível lógico baixo, e permanecerá baixo até que o trigger seja disparado novamente. A equação da temporização do pulso em nível logico é apresentado pela equação (10).

$$t = 1{,}1RC \tag{10}$$

CAPÍTULO 3

3. Métodos

3.1. Tanque de Combustível

Como dito na introdução o capacitor utilizado como sensor deve ficar na parte externa do tanque, pois devido a questões de segurança não se permite que fios saiam de dentro do tanque, e o posicionamento dos capacitores externo ao tanque foram favorecidos pelo aspecto construtivo do tanque, que possui três cavidades na parte inferior do tanque, apresentado pelas Figuras 4 e 5.

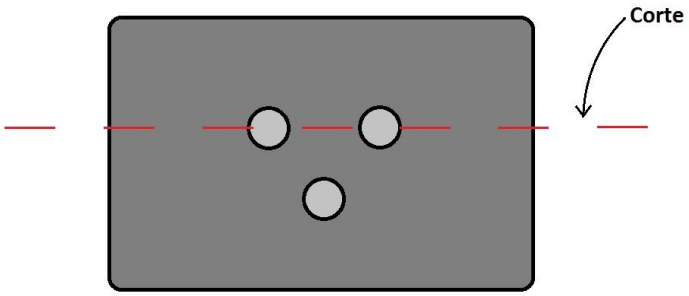

Figura 4: Visão do fundo do tanque.

As medidas das cavidades do tanque foram realizadas por aproximação, resultando em aproximadamente 7cm de profundidade e 3cm de diâmetro, e uma distância entre as cavidades de 3cm.

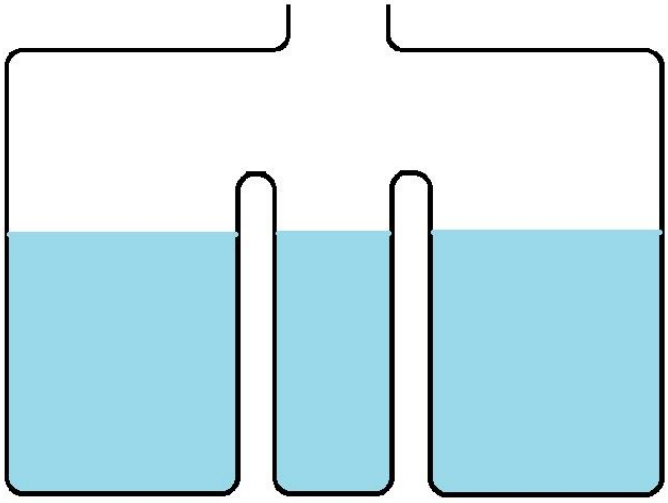

Figura 5: Visão do corte no tanque apresentado pela Figura 4.

3.2. Circuito de Detecção de Nível

Para a detecção de nível foi proposto a utilização do circuito integrado 555 na configuração monoestável em conjunto com os sensores capacitivos, assim a cada nível do liquido o circuito apresentará um tempo em nível logico alto proporcional ao nível de combustível presente no tanque. Portanto, montou-se três osciladores monoestáveis, um para cada capacitância disposta entre as três cavidades do tanque.

Para realizar a contagem de tempo em nível logico alto utilizou-se a contagem de tempo por software. Uma vez com a janela de tempo é definida pela capacitância do tanque, assim gerando uma determinada medida de tempo equivalente ao nível de combustível no tanque, que entrara em um dos pinos do microcontrolador do Arduino UNO, e através da multiplexação realiza-se a leitura dos três sensores.

Uma vez que calculado o nível do tanque o microcontrolador retorna a saída para três LEDs conectados ao microcontrolador por um conjunto de potência, formado por transistores, para que o microcontrolador não forneça mais corrente do que é capaz. Poderia utilizar mais LEDs, porem o carro possui uma limitação de consumo de corrente, portanto preferiu-se utilizar apenas seis, cinco para indicação do volume e um para indicação de alerta. Pode-se observar o esquema elétrico do circuito de detecção na Figura 6.

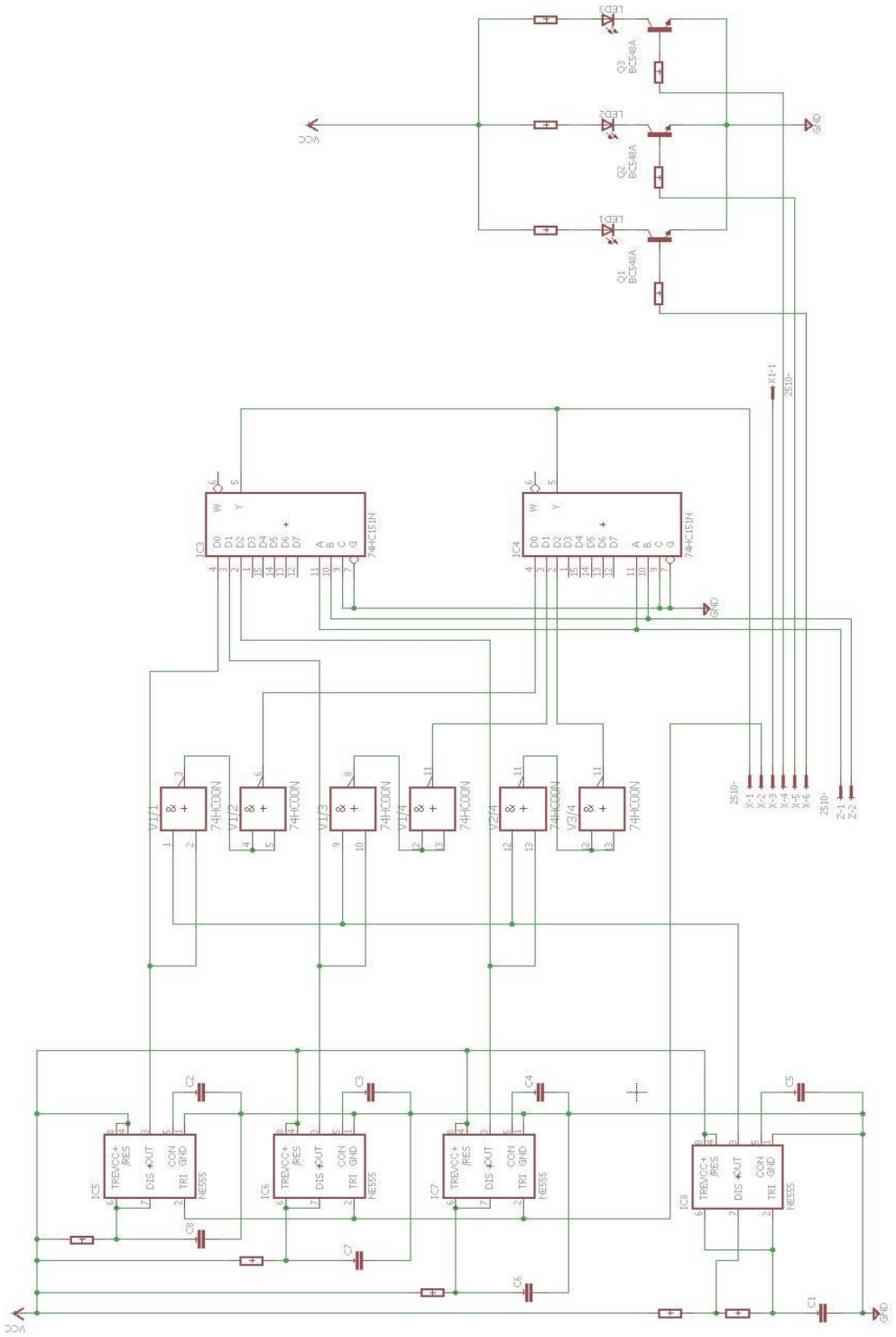

Figura 6: Esquema elétrico do circuito de detecção de nível.

3.3. Fluxograma do Programa de Cálculo do Nível

Para o fluxograma de cálculo de nível devemos considerar duas hipóteses, a primeira quando o carro esteja parado, assim devemos considerar a possibilidade de abastecimento do carro, a segunda, o carro em movimento, assim devemos considerar que não é possível o aumento de combustível no tanque, portanto, podemos eliminar ruídos na medição caso a medida colhida apresente um nível de combustível maior que as medidas anteriores, outra consideração importante é a possibilidade de vazamento, assim deve-se verificar caso as medidas sequencialmente resultem em níveis de combustíveis menores do que se espera. O fluxograma do programa desenvolvido para a estimação do nível de combustível no tanque pode ser visto na Figura 7.

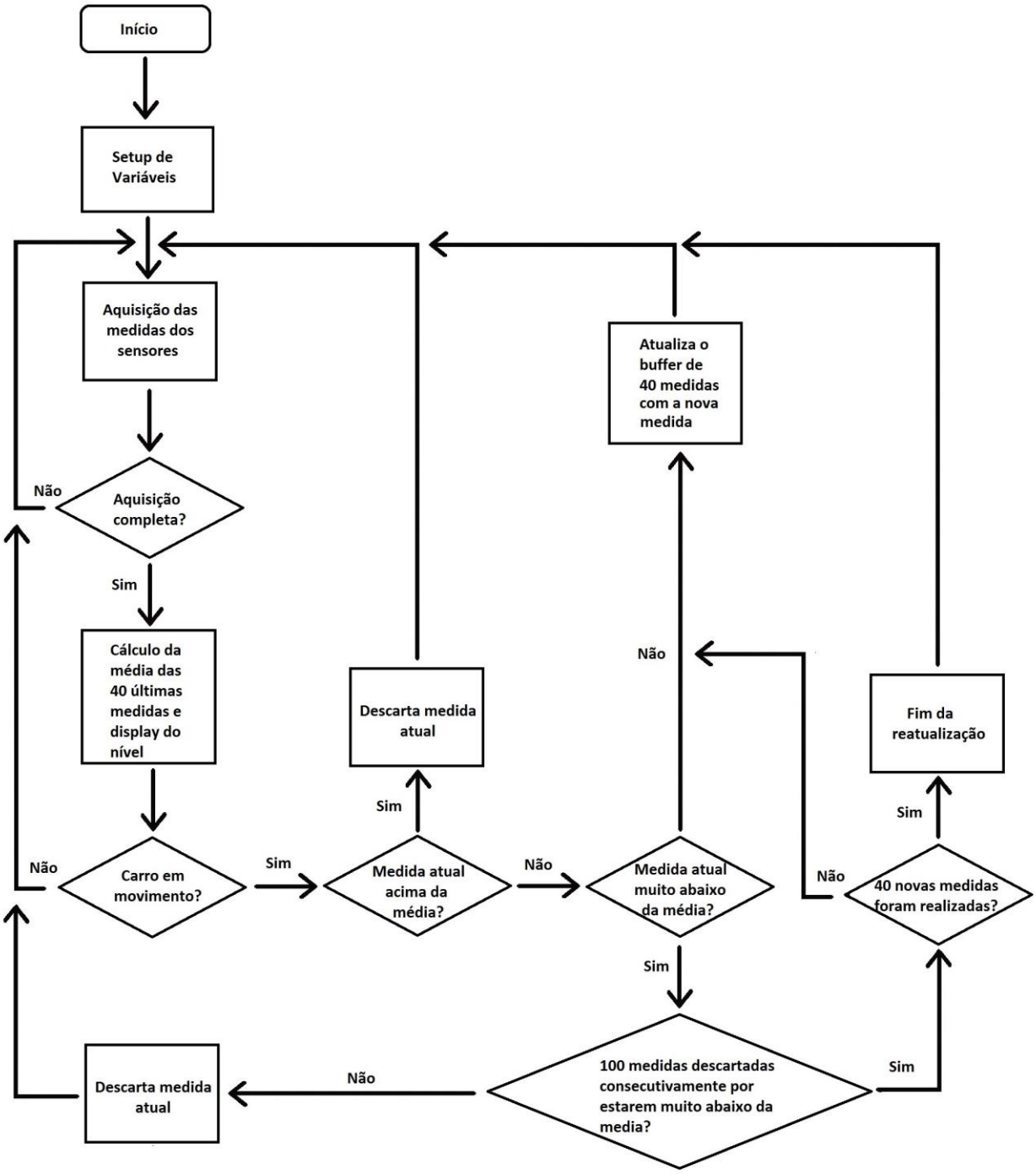

Figura 7: Fluxograma do programa de detecção de nível.

CAPÍTULO 4

4. Resultados e Discussão

4.1. Capacitância

Para o cálculo da capacitância dos sensores, por simplificação levou-se em consideração o formato mais simples de capacitor, o de placas paralelas. Com isso calculou-se as capacitâncias máximas e mínima as quais correspondem respectivamente ao tanque cheio e ao tanque vazio. Realizou-se uma pesquisa sobre o dielétrico da gasolina, encontrou-se o valor do dielétrico relativo, o qual possui valor igual a 2, assim com o tanque cheio a capacitância será o dobro de quando o tanque estiver vazio.

O cálculo realizado para a capacitância utilizou-se as seguintes medidas para o cálculo da área: 7cm x 2,5cm; e uma distância entre as placas de 3cm. Para o cálculo desprezou-se o dielétrico do tanque. Resultando assim para o tanque cheio: 1,0325pF; e para o tanque vazio 0,516pF.

Para estimar a capacitância mínima para um bom funcionamento do programa e circuito, deve-se fazer algumas considerações, como um bom valor de tempo para a janela de tempo de tal modo que gere valores de tempo significativos para se distinguir o ruído e possuir uma boa resolução de variação. Então, levando-se em consideração o valor máximo de clock do microcontrolador utilizado, que é 16MHz, podemos contar o tempo em microssegundos, que fornece um bom fundo de escala para realizar cálculos mais precisos e eliminar ruídos da capacitância fazendo um simples cálculo da média dos valores dos sensores.

Estimando uma resolução de contagem mínima de 5000 microsegundos para o tanque vazio, portanto 10000 para o tanque cheio, e considerando a frequência de variação do nível do tanque seja consideravelmente menor que a de geração de tempo, uma vez que a variação mecânica de nível é mais lenta que a contagem mínima de tempo, temos os tempos das janelas: 5ms para o

tanque vazio e 10ms para o tanque cheio. Com esses valores, podemos recorrer a equação 10, e estimar a capacitância mínima (caso para o tanque vazio, pois quando cheio a capacitância sempre será maior) para o devido funcionamento do sistema. A Tabela 1 apresenta os valores de capacitância mínima para o devido funcionamento do circuito.

Resistencia	**Capacitância**
47kΩ	96,71nF
100kΩ	45,45nF
220kΩ	20,66nF
470kΩ	9,67nF
8.8 GΩ	0,516pF

Tabela 1: Valores mínimos de capacitância dos sensores.

Como podemos observar na Tabela 1, os valores de capacitância estão cerca de dez mil vezes superior ao calculado para o capacitor de placas paralelas máximo (utilizando-se valores de resistências a baixo de mega) capaz de encaixar nas cavidades do tanque (último valor da tabela). Entretanto, existe uma solução simples, pode-se utilizar um maior valor de tempo para as estimações para dez vezes maiores, assim pode-se utilizar uma resistência de 880kΩ para a capacitância mínima do tanque. Uma outra solução seria utilizar um circuito multiplicador de capacitância e assim escalar para um valor de capacitância funcional.

4.2. Circuito Detector

Realizou-se testes no circuito montado no protoboard, o circuito apresentou a resposta esperada. Entretanto, o protoboard apresenta capacitâncias parasitas, além de trilhas defeituosas

que causaram em ocasiões oscilações indesejadas, offset de tensão e atenuação de sinal. Para efeito de teste utilizou-se capacitores eletrolíticos de 4,7µF para simular os sensores.

4.3. Simulação da Detecção de Nível

O software desenvolvido não chegou a ser posto a teste de pratica, entretanto realizou-se simulações do software utilizando-se a ferramenta MATLAB.

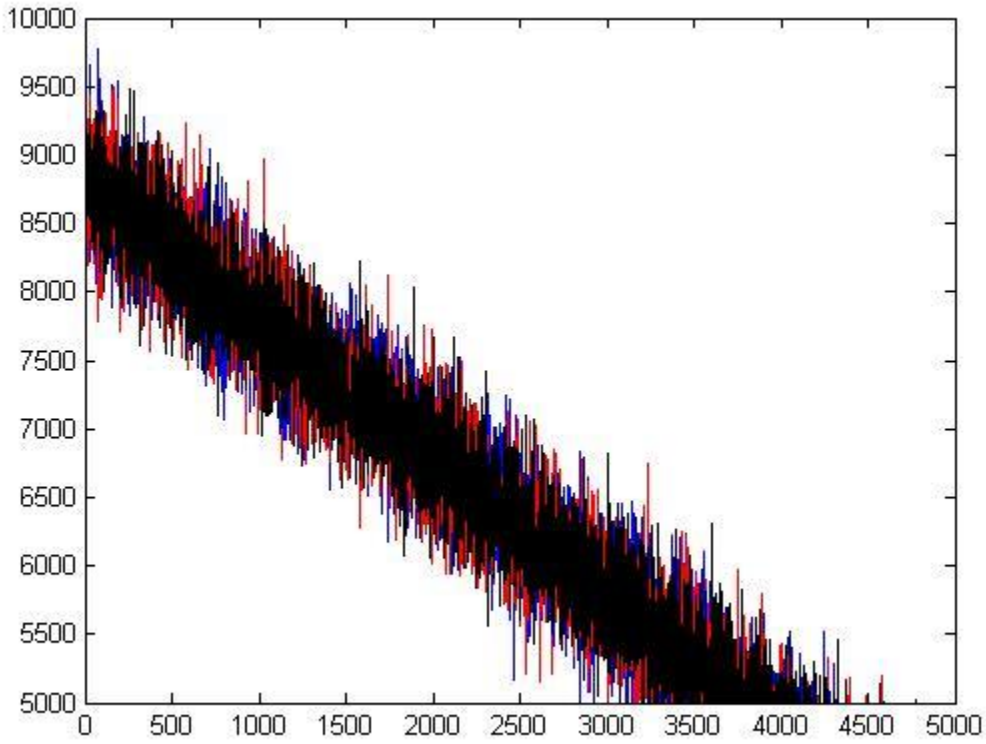

Figura 8: Medidas dos sensores individuais.

Pode-se observar na Figura 8 os valores de tempo medidos em cada medida para cada sensor, cada conjunto de medidas de um determinado sensor apresentado por uma cor diferente,

azul, vermelho e preto. A Figura 8 apresenta os valores medidos de capacitância incluindo ruídos na capacitância e a variação de nível devido à movimentação do carro.

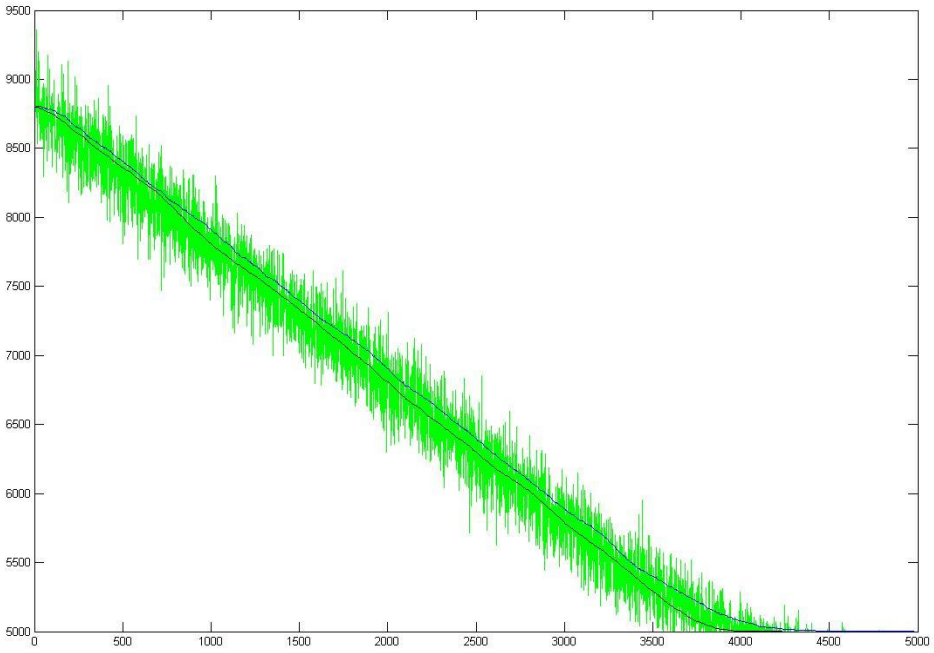

Figura 9: Nível estimado.

A Figura 9 apresenta em verde a média dos três sensores a cada medida realizada, em preto observa-se o nível estimado pelo algoritmo de truncamento pela média desenvolvido, em azul observa-se um algoritmo simples, um filtro de média. Para uma melhor visualização pode-se observar na Figura 10, um zoom da Figura 9, as diferentes respostas.

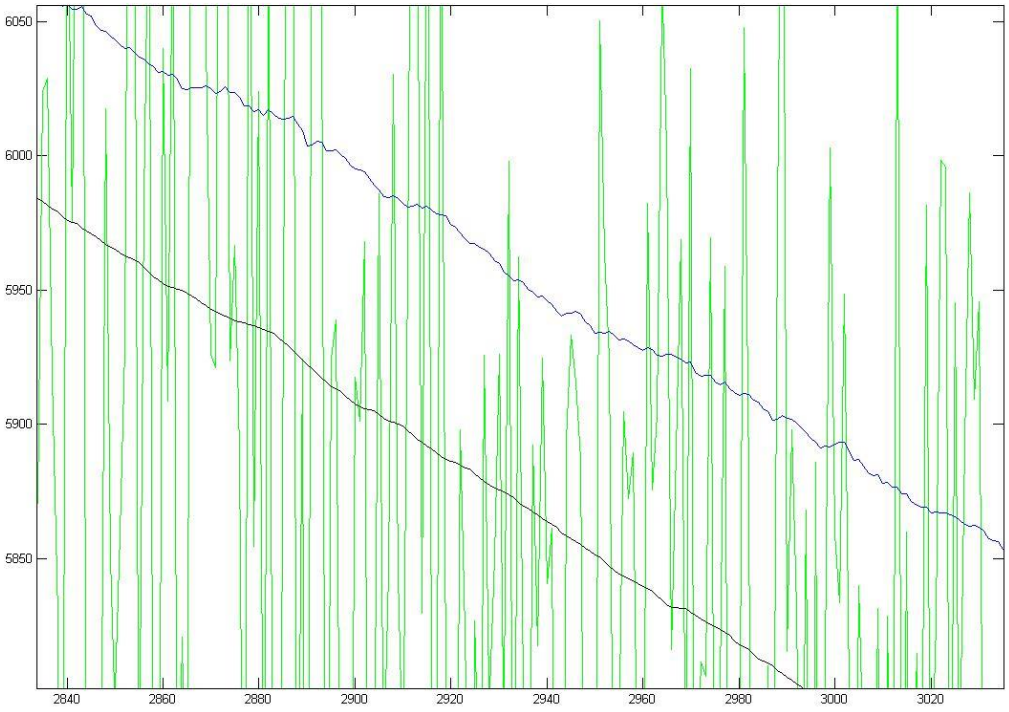

Figura 10: Truncamento pela Média x Filtro de Média.

Em preto observa-se uma medição de nível claramente mais suave, evitando variações indesejadas, entretanto o algoritmo apresenta uma resposta de nível ligeiramente mais baixa que a resposta apresentada pelo Filtro de Média, porém o algoritmo desenvolvido apresenta uma resposta mais realista do nível verdadeiro.

CAPÍTULO 5

5. Conclusão

O circuito apresentou a resposta esperada e satisfatória atendendo às expectativas de resposta, portanto pode-se dar continuidade ao projeto utilizando o esquema proposto por este trabalho. Para a continuidade do projeto deve-se recalcular as capacitâncias utilizando equações para o formato adequado da cavidade do tanque, e com isso ajustar o software para valores de janela de tempo para que se possa utilizar valores de resistências menores, ou analisar a possibilidade de se projetar um multiplicador de capacitância que atenda aos requisitos mínimos desse projeto.

Deve-se colocar o software de detecção em teste e analisar o seu comportamento e calibrar seus valores para o devido funcionamento do sistema de detecção de nível de combustível.

ANEXO

Anexo 1

Os programas abaixo foram utilizados para a estimação de nível utilizando o microcontrolador Arduino UNO e a simulação do algoritmo utilizando o software MATLAB.

Nivel.ino

```
int trigger = 12;
int movimento = 11;
int led1 = 10;
int led2 = 9;
int led3 = 8;
int led4 = 7;
int mux0 = 6;
int mux1 = 5;

int ledR = 4;

unsigned long started_at_time = 0;
unsigned long mean, H_mean, aux;
unsigned long Sensors_response[3], Means[200];
unsigned long Aux = 1000;
unsigned int counter = 0;
int cnt = 0;
int ok = 0;
int i = 0;
int index = 0;
int fail = 0;
int verif = 0;
int dspl = 0;

void interrupt_process();
void mean_function();
void higher_mean();
void higher_mean_verification();
void Trigger();
void mux();
```

```
  void leds_display();

void setup() {

  Serial.begin(9600);

  pinMode(trigger,OUTPUT);
  pinMode(movimento,INPUT);
  pinMode(led1,OUTPUT);
  pinMode(led2,OUTPUT);
  pinMode(led3,OUTPUT);
  pinMode(led4,OUTPUT);
  pinMode(ledR,OUTPUT);
  pinMode(mux0,OUTPUT);
  pinMode(mux1,OUTPUT);

  attachInterrupt(0,interrupt_process,FALLING);       //interrupcao no pino 2

  digitalWrite(trigger, HIGH);

  digitalWrite(mux0, LOW);
  digitalWrite(mux1, LOW);

  digitalWrite(led1, LOW);
  digitalWrite(led2, LOW);
  digitalWrite(led3, LOW);
  digitalWrite(led4, LOW);
  digitalWrite(ledR, LOW);

  unsigned long time = micros();

  Trigger();

}

void loop() {
  unsigned long time = micros();

  if(ok == 1) {
    higher_mean();

    leds_display();

    if(digitalRead(movimento)) {
      higher_mean_verification();
    }
    else {
      Means[index] = mean;

      dspl = 1;

      index++;
    }

    if(dspl == 1){
      dspl = 0;
    }
```

```
    else{
      if(index > 1){
        Means[index] = Means[index - 1];

        index++;
      }
      else{
        Means[index] = Means[199];

        index++;
      }
    }

    if(index == 200) {
      index = 0;
    }

    ok = 0;

    Trigger();
  }

}

void interrupt_process() {
  Sensors_response[i] = micros() - started_at_time;

  i++;
  cnt++;

  mux();

  if(i = 3){
    mean_function();

    cnt = 0;
    mux();
    i = 0;
    ok = 1;
  }
  else{
    Trigger();
  }
}

void mean_function() {
  mean = 0;

  for(int j = 0; j<3; j++)
    mean = mean + (1/3)*Sensors_response[j];
}

void higher_mean() {
  H_mean = 0;

  for(int k = 0;k<200;k++)
     H_mean = H_mean + (1/200)*Means[k];
```

```
}

void higher_mean_verification() {
  aux = H_mean - Aux;

  if(mean < H_mean) {
      if(mean > aux) {
          dspl = 1;

          Means[index] = mean;

          index++;

          fail = 0;
      }
      else {
          fail++;
      }
  }

  if(fail > 20) {
      Means[index] = mean;
      dspl = 1;

      index++;

      verif++;

      if(verif == 200) {
          fail = 0;
          verif = 0;
      }
  }
}

void Trigger() {
  started_at_time = micros();

  digitalWrite(trigger, LOW);
  delayMicroseconds(100);
  digitalWrite(trigger, HIGH);
}

void mux() {
  if(i == 0) {
    digitalWrite(mux0, LOW);
    digitalWrite(mux1, LOW);
  }

    if(i == 1) {
    digitalWrite(mux0, HIGH);
    digitalWrite(mux1, LOW);
  }

    if(i == 2) {
    digitalWrite(mux0, LOW);
    digitalWrite(mux1, HIGH);
```

```
  }
}

void leds_display() {
  if(H_mean > 8750){
    digitalWrite(led1, HIGH);
    digitalWrite(led2, HIGH);
    digitalWrite(led3, HIGH);
    digitalWrite(led4, HIGH);
  }
  if(H_mean < 8750 && H_mean > 7500){
    digitalWrite(led1, LOW);
    digitalWrite(led2, HIGH);
    digitalWrite(led3, HIGH);
    digitalWrite(led4, HIGH);
  }
  if(H_mean < 7500 && H_mean > 6250){
    digitalWrite(led1, LOW);
    digitalWrite(led2, LOW);
    digitalWrite(led3, HIGH);
    digitalWrite(led4, HIGH);
  }
  if(H_mean < 6250 && H_mean > 5650){
    digitalWrite(led1, LOW);
    digitalWrite(led2, LOW);
    digitalWrite(led3, LOW);
    digitalWrite(led4, HIGH);
  }
  if(H_mean < 5650){
    digitalWrite(led1, LOW);
    digitalWrite(led2, LOW);
    digitalWrite(led3, LOW);
    digitalWrite(led4, LOW);
  }

//-------------------------------------------------------------------------
  if(H_mean > 5650){
    digitalWrite(ledR, LOW);
  }
  if(H_mean < 5250){
    digitalWrite(ledR, HIGH);
  }
}
```

Sim.m

```
t1 = 50*randn(1,5000);
t2 = 50*randn(1,5000);
t3 = 50*randn(1,5000);

t = 0:1:4999;
```

```
x1 = t1 + t;
x2 = t2 + t;
x3 = t3 + t;

p1 = 300*randn(1,5000) + 3800;
p2 = 300*randn(1,5000) + 3800;
p3 = 300*randn(1,5000) + 3800;

Pcnte = 5000;

for i=1:5000
    y1(i) = p1(i) - x1(i);
    y2(i) = p2(i) - x2(i);
    y3(i) = p3(i) - x3(i);
end

for i=1:5000
    if (y1(i) > 5000)
        y1(i) = 5000;
    end
    if (y2(i) > 5000)
        y2(i) = 5000;
    end
    if (y3(i) > 5000)
        y3(i) = 5000;
    end
    if (y1(i) < 0)
        y1(i) = 0;
    end
    if (y2(i) < 0)
        y2(i) = 0;
    end
    if (y3(i) < 0)
        y3(i) = 0;
    end
end

for i=1:5000
    y1(i) = y1(i) + Pcnte;
    y2(i) = y2(i) + Pcnte;
    y3(i) = y3(i) + Pcnte;
end

%--------------------------------------------------------------------------

H_mean = 3800 + Pcnte;
aux = 230;

index = 1;
fail = 0;
verif = 0;
dspl = 0;

mean = [];
```

```
Means = [];
R_mean = [];
dspl = 0;

N = 200;

for i=1:N
    Means(i) = 3800 + Pcnte;
end

for i=1:5000
    mean(i) = (1/3)*(y1(i) + y2(i) + y3(i));

    H_mean = 0;

    for k=1:N
        H_mean = H_mean + (1/N)*Means(k);
        R_mean(i) = H_mean;
    end

    Ax = H_mean - aux;
    if(mean(i) < H_mean)
        if(mean(i) > Ax)
            Means(index) = mean(i);
            dspl = 1;

            index = index + 1;

            fail = 0;
        else
            fail = fail + 1;
        end
    end

    if(fail > 20)
        Means(index) = mean(i);
        dspl = 1;

        index = index + 1;

        verif = verif + 1;
        if(verif == N)
            fail = 0;
            verif = 0;
        end
    end

    if(dspl == 1)
        dspl = 0;
    else
        if(index > 2)
            Means(index) = Means(index - 1);

            index = index + 1;
```

```
        else
            Means(index) = Means(100);

            index = index + 1;
        end
    end

    if(index > N)
        index = 1;
    end
end

%-------------------------------------------------------------------------

index = 1;

M_mean = [];

for i=1:N
    Means(i) = 3800 + Pcnte;
end

for i=1:5000
    mean(i) = (1/3)*(y1(i) + y2(i) + y3(i));

    H_mean = 0;

    Means(index) = mean(i);

    index = index + 1;

    for k=1:N
        H_mean = H_mean + (1/N)*Means(k);
        M_mean(i) = H_mean;
    end

    if(index > N)
        index = 1;
    end
end

figure(1);
plot(t,y1,'blue')
hold on
plot(t,y2,'red')
hold on
plot(t,y3,'black')

figure(2);
plot(t,mean,'green')
hold on
plot(t,R_mean,'black')
hold on
plot(t,M_mean,'blue')
```