RESUMO

O projeto Automares, referente ao terceiro período do curso de Engenharia de Computação da Pontifícia Universidade Católica do Paraná, propõe o desenvolvimento de um varal automatizado capaz de proteger as roupas da chuva através de previsões climáticas.

Palavras-chave: projeto, varal, automatizado, chuva, roupas.

SUMÁRIO

1 INTRODUÇÃO

O projeto Automares consiste na automatização do processo de recolher e estender as roupas baseado nas condições climáticas. A partir de leituras feitas por uma pequena estação meteorológica - que conta com sensores de umidade, pressão, temperatura e de chuva – um algoritmo tomará a decisão de estender ou recolher as roupas. Tal decisão também pode ser tomada pelo usuário, com exceção do momento em que há possibilidade ou presença de chuva onde o usuário deixa de ter permissão para estender a estrutura.

O diferencial da proposta do grupo é a previsão de chuva para que a roupa seja recolhida antes de se molhar, pois há projetos similares que contam apenas com a tomada de decisão a partir do momento em que há percepção da chuva.

2 OBJETIVOS

Para uma boa compreensão do projeto é necessário conhecer suas principais características e todos os requisitos que o mesmo deverá atender.

2.1 OBJETIVO GERAL

Desenvolver um projeto para os programas de aprendizagem das disciplinas de Física III, Resolução de Problemas em Engenharia I e Sistemas Digitais I que consista em um varal de roupas automatizado capaz de evitar prever a possibilidade de chuva e evitar que as roupas que estão secando tornem a se molhar.

2.2 OBJETIVOS ESPECÍFICOS

Abaixo serão listados os objetivos que regeram o desenvolvimento do projeto:

- Desenvolver um varal contendo uma parte móvel, onde as roupas serão fixadas, e outra fixa que servirá de sustentação.
- Ligar a parte móvel do varal a um motor;
- Controlar o motor para deslocar a parte móvel da estrutura da maneira desejada;
- Determinar grandezas físicas determinantes para a previsão meteorológica;
- Criar algoritmo de previsão meteorológica;
- Baseado nas respostas do algoritmo meteorológico tomar as devidas decisões relacionadas ao controle da estrutura móvel;
- Desenvolver interface para interação com usuário.
- Programar as interferências que o usuário pode infligir sobre o sistema.

3 MATERIAIS UTILIZADOS

- 1 Arduino Uno;
- 1 Carrinho de Impressora com Motor DC;
- 1 CI L293D;
- 22 Bornes;
- 2 Push-Buttons;
- 1 Barra de Pinos;
- 3 Plugs Jack;
- 2 Conectores Jack;
- 10 Resistores de 10KΩ;
- 2 Resistores de 330Ω;
- 1 Resistor de 220Ω;
- 1 Resistor de 100Ω;
- 1 LED de Alto Brilho de 3mm;
- 2 LEDs Comuns de 5mm;
- 2 Fins-de-Curso Mecânicos (Sensor de Toque);
- 1 Capacitor de 100nF;
- 1 Buzzer contínuo de 3V;
- 1 Caixa de Plástica PB119/2-TE;
- Placas de Fenolite;
- 1 Sensor de Pressão e Temperatura BMP085;
- 1 Sensor de Umidade e Temperatura DHT22;
- Madeira;
- Parafusos;
- Tinta em Spray Preta;
- Trilho de Gaveta;
- Cabides.

4 DESCRIÇÃO GERAL

4.1 HISTÓRIA DO PROJETO

Inicialmente a ideia do projeto foi de automatizar um varal utilizando somente um sensor de chuva. Seguindo a sugestão do professor Afonso Miguel (Resolução de Problemas em Engenharia I) foram adicionados os sensores de pressão, temperatura e umidade para que momentos antes de chover a estrutura fosse recolhida, ao invés de recolher-se somente ao chover.

Após revisar e estipular os conceitos do projeto o mesmo foi oficializado em forma de plano de trabalho e entregue ao professor da disciplina de Resolução de Problemas em Engenharia como proposta de trabalho para o semestre.

4.1.1 ELETRÔNICA

O projeto eletrônico iniciou-se com a definição dos módulos necessários para viabilizar todas as funcionalidades propostas. Foram eles:

- Sensores;
- Interface de Usuário;
- Controle de Motores;

Devido ao pouco conhecimento de eletrônica e à simplicidade do circuito, o passo tomado em seguida foi projetar a interface de interação com o usuário. (Figura 1.)

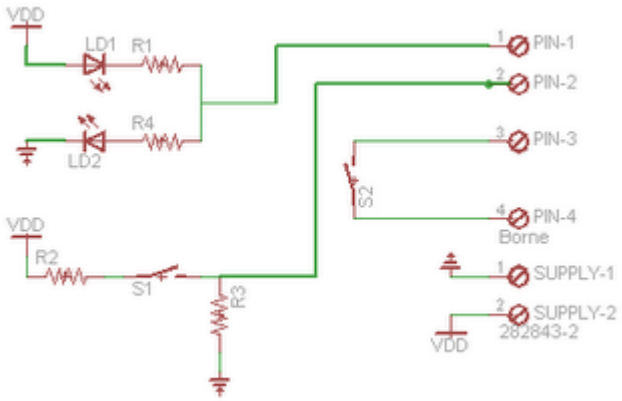

Figura 1

Inicialmente este circuito propunha utilizar uma única porta do Arduino para controlar dois leds indicativos (LD1 e LD2), um push-button (S1) para controle e uma chave mecânica (S2) para cortar a alimentação do sistema.

Como este circuito era uma peça fundamental para o andamento do projeto, foi implementado apenas no momento de integração, porém foram encontrados alguns problemas. Aparentemente o mau funcionamento provinha de algum curto-circuito, pois os resultados esperados foram observados com exceção de um dos leds que acabou queimando. Devido à falta de tempo e à pretensão de implementar um buzzer, o circuito foi abandonado e um novo foi projetado com o mesmo intuito. Além disto foi removida a chave mecânica pois o projeto não previa construção da fonte de alimentação, logo o gerenciamento da energia seria feito pela fonte externa, e não pelo próprio sistema. (Figura 2.)

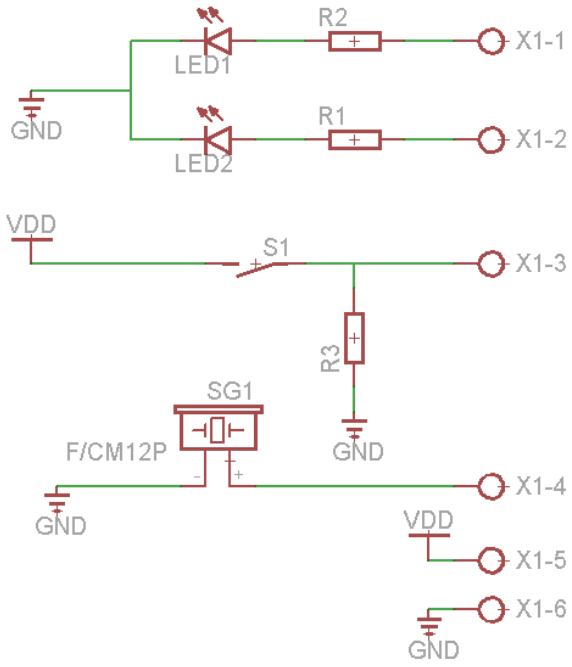

Figura 2

O passo seguinte foi determinar o modo de construção da ponte-h. Foi pensando em construí-la através de transistores de potência ou utilizando CIs. Por sugestão dos professores Ivan Chueiri e Afonso Miguel, foi adotado o CI L298N, e então desenhado o esquema de componentes. (Figura 3.)

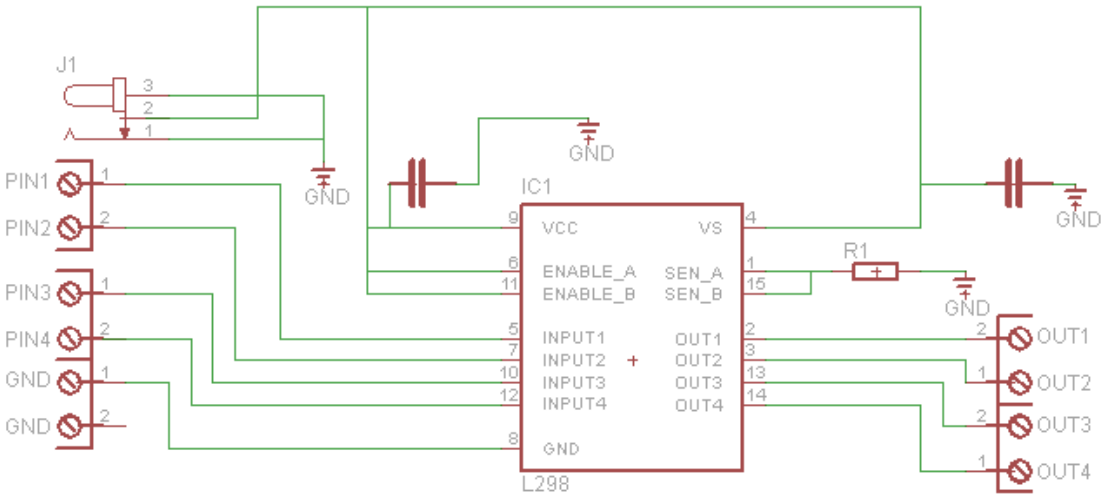

Figura 3

Após construir a PCI observaram-se problemas intermitentes, provavelmente relacionados às vias que em alguns casos estavam interrompidas ou em curto, algumas estavam muito finas, outras muito próximas (observar figuras 4 e 5). Feito um reparo o circuito passou a funcionar de maneira adequada por certo período, porém voltou a apresentar irregularidades.

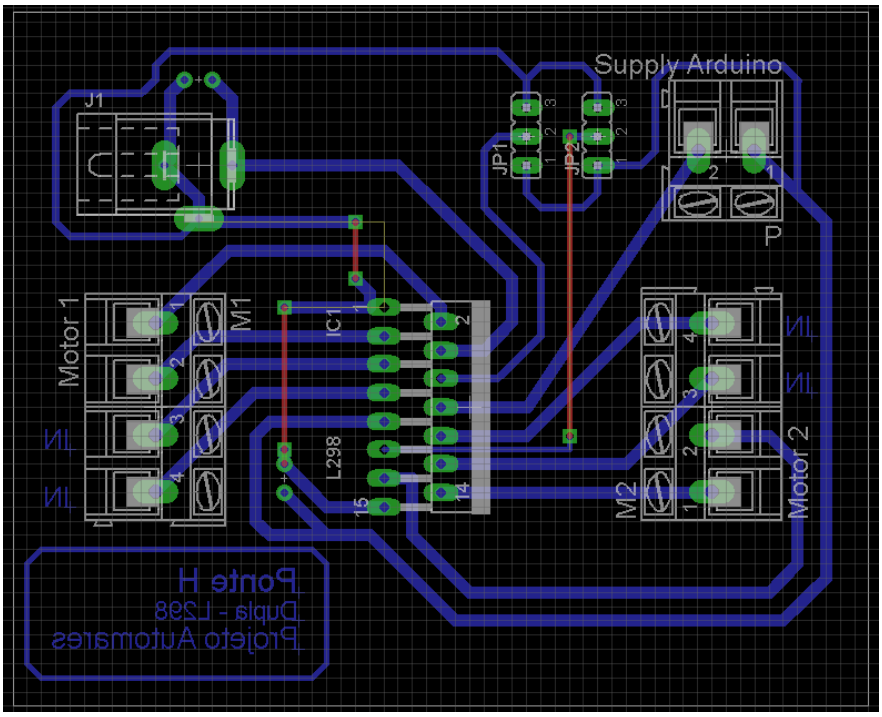

Figura 4

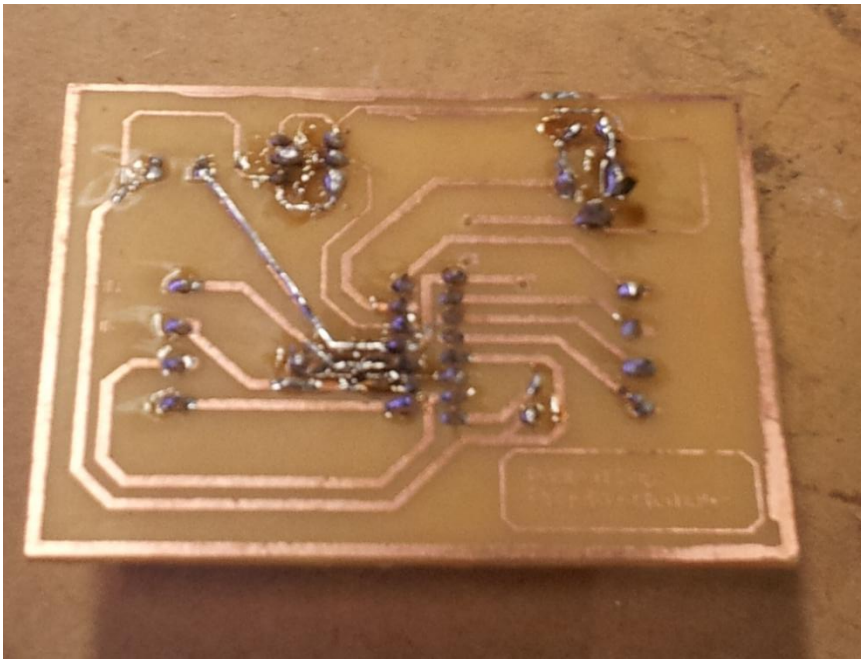

Figura 5

Levando em consideração os problemas com a confecção do circuito, o desenho da board foi refeito, porém ainda houve problemas. Tais problemas podem ser atribuídos a erros de utilização.

Visto que o prazo de conclusão do controle de motores estava para expirar utilizou-se um circuito de ponte-h, baseado no CI L293D, construído anteriormente pelo integrante Guilherme. O emprego do mesmo ofereceu os resultados desejados.

Pensando na coleta de dados para o estudo meteorológico foi construído um sensor de chuva, para que fosse possível separar as leituras de dias de chuva.

Inicialmente continha apenas vias que seriam curto-circuitadas pelas gotas d'água, entradas de alimentação e um resistor de pull-down de 10KΩ. Ao implementá-lo notou-se ruído excessivo, e por isso foi adicionado um capacitor de 100nF entre o GND e o pino de leitura, ao verificar as tensões de entrada e saída com o multímetro e observar a saída nas impressões do Arduino era possível observar os resultados esperados, entretanto ao retirar o multímetro voltava a haver interferência nas impressões. A partir disso deduziu-se que a resistência interna do multímetro estava interferindo no circuito, e após alguns testes foi possível confirmar que era necessário aumentar o resistor de pull-down, que passou a ter ser de 1MΩ. (Figura 6.)

Figura 6

Com o intuito de organizar os módulos do projeto e aumentar o número de conexões ao VCC e ao GND foi desenvolvido um shield para converter as saídas do Arduino em bornes. (Figura 7.)

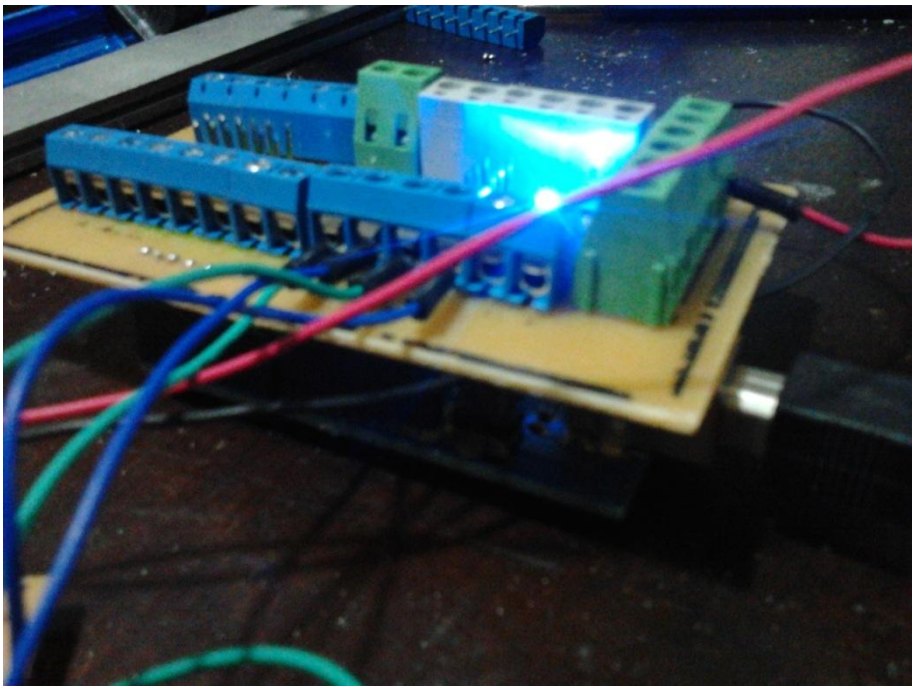

Figura 7

Finalizando a parte eletrônica foi desenvolvido o circuito responsável por ligar os fins-de-curso ao Arduino. (Figura 8.)

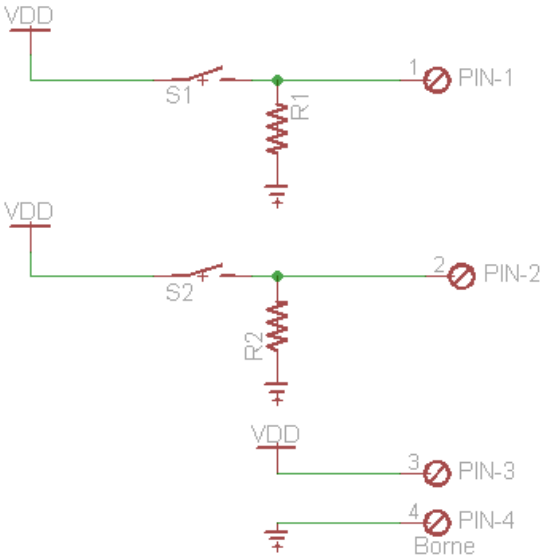

Figura 8

4.1.2 MECÂNICA

O projeto mecânico partiu da definição dos conceitos e formas básicas da estrutura, em forma de CAD. (Figura 9.)

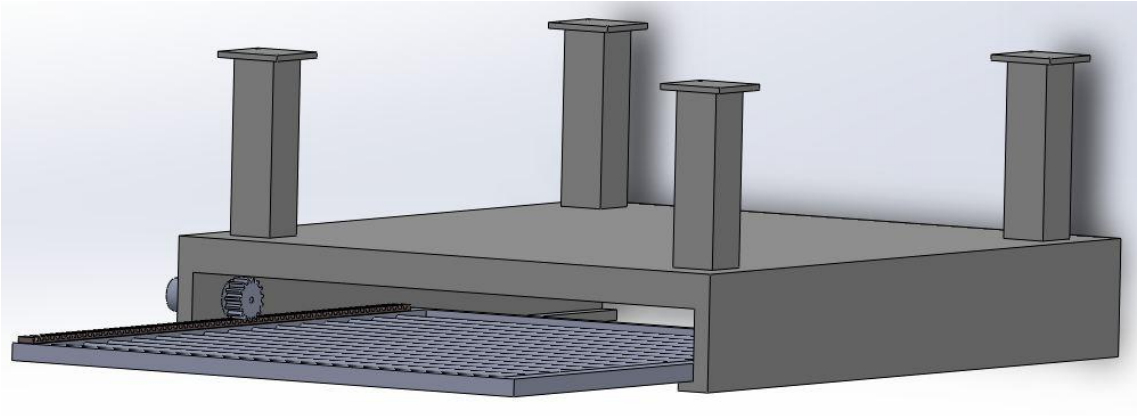

Figura 9

Em seguida o grupo discutiu com o professor Gil M. Jess as possíveis formas de movimentar a parte onde seriam penduradas as roupas. Após discutir várias ideias, foi decido utilizar trilhos de gaveta. (Figura 10.)

Figura 10

O passo seguinte consistiu em confeccionar a estrutura base do varal. (Figuras 11.)

Figura 11

Após finalizar a estrutura base, e definido o modo como a parte móvel deslizaria sobre a fixa, a mesma foi construída utilizando um gaveta de calças. Contudo, devido a um erro de corte o tamanho da estrutura móvel não era compatível com o da fixa.

Sendo assim a gaveta de calças foi descartada e a nova parte móvel foi confeccionada construindo uma moldura de madeira onde foram encaixadas varetas metálicas, retiradas de cabides. (Figura 12.)

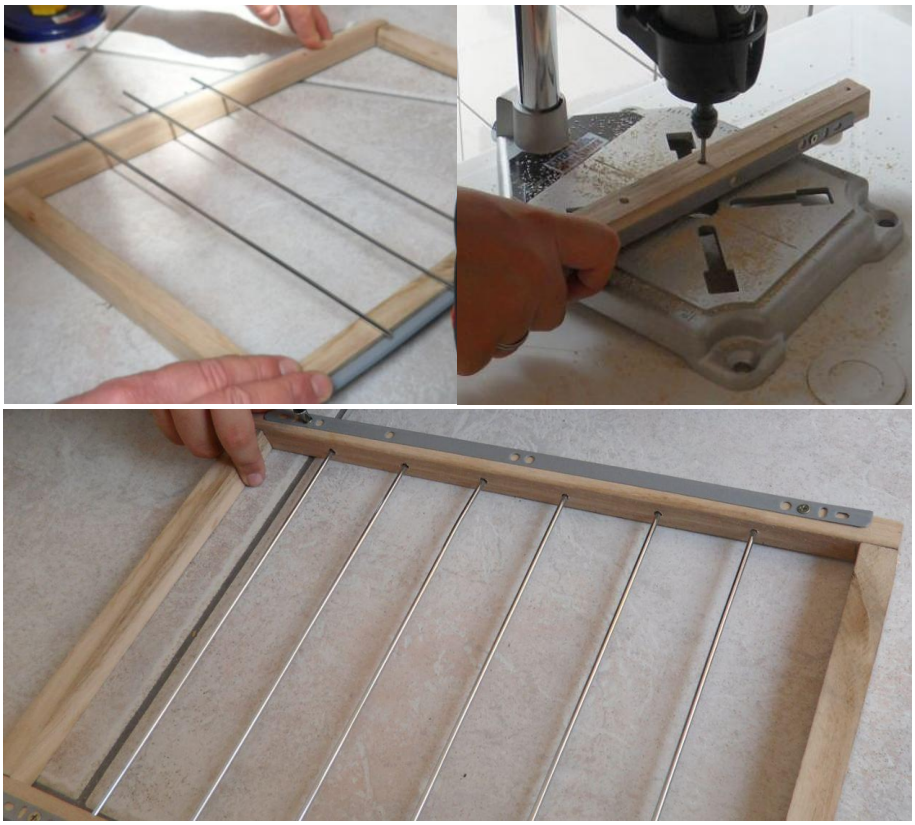

Figura 12

Para mover o varal é necessário haver contato entre o motor e a parte móvel. Num primeiro momento a ideia era de utilizar um conjunto pinhão/cremalheira, contudo devido ao alto custo de fabricação do mesmo foi necessário buscar um meio alternativo. Para a solução deste problema foi adaptado um carrinho de impressora onde a parte plástica, que na impressora funciona como o suporte para os cartuchos, foi presa à parte móvel. Já a parte metálica foi fixada à estrutura fixa do varal. (Figura 13.)

Figura 13

4.1.3 SOFTWARE

O passo inicial para o desenvolvimento do software foi a criação de um fluxograma que retratasse as principais características que o programa deveria ter. (Figura 13.)

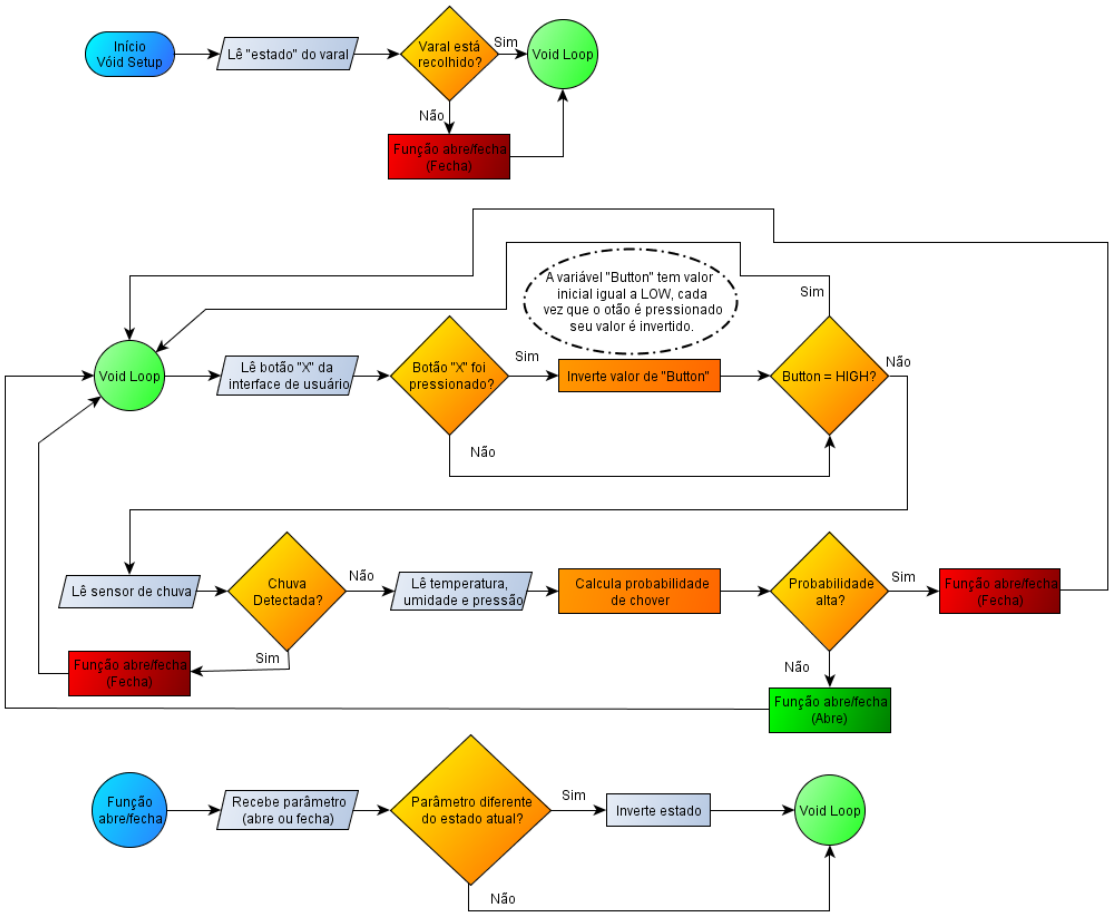

Figura 14

Ao receber os sensores de pressão, temperatura e umidade (DHT22 e BMP085) foi iniciado o processo de desenvolvimento do software para o Arduino, onde a primeira etapa foi caracterizada pela criação e teste dos softwares de leitura dos respectivos sensores.

Em seguida iniciou-se o desenvolvimento de um programa, baseado em Processing, que seria responsável por receber os dados das leituras climáticas, provenientes dos sensores conectados ao Arduino, diferenciar os dados de quando houvesse chuva e armazená-los no disco rígido de um computador para que baseado nesses dados fosse possível desenvolver o algoritmo de previsão meteorológica. (Figura 14.)

Figura 14

Mesmo sendo concluído este programa não foi utilizado por dois motivos, o primeiro deles está ligado à falta de tempo para tais leituras e o segundo motivo está ligado à possibilidade de encontrar estes dados no site do INMET (Instituto Nacional de Meteorologia).

Após definir o algoritmo de previsão meteorológica foi necessário utilizar seu resultado para controlar os demais ramos do software, foi então implementado o controle de motores e controle do usuário, além de um sensor prioritário – o sensor de chuva – que é a forma de mais alta preferência sobre o controle da estrutura.

4.1.4 METEOROLOGIA

Inicialmente a ideia que se tinha para fazer a previsão da chuva era desenvolver uma equação matemática em função de três grandezas físicas – pressão, temperatura e umidade – que aplicadas à equação gerariam um valor representando a probabilidade de chuva. Em conversa com o professor Gil M. Jess foi possível verificar a impossibilidade de fazê-lo, pois seria necessário um alto processamento, coisa que o Arduino não oferece.

Uma segunda possibilidade para a previsão de chuva era a de criar intervalos de comparação para cada grandeza obtida. Tal intervalo foi obtido através de uma tabela encontrada no site do INMET, que continha dados diários relacionados às condições climáticas da região de Curitiba incluindo a quantidade de chuva, os quais foram separados em dois grupos os que apresentaram ou não chuva.

O algoritmo de previsão acabou constituído da seguinte forma, para os dois grupos foram calculados os desvios padrão e as médias de pressão, umidade e temperatura. Estes dados foram utilizados para definir intervalos para duas funções do algoritmo, a PodeChover e a NaoDeveChover, definidos da média subtraída do desvio padrão até a média somada pelo desvio padrão para todas as três leituras previstas.

Para tomar a decisão de recolher a estrutura é necessário que o resultado das duas funções criadas seja condizente, ou seja, que a função PodeChover retorne verdadeiro e a NaoDeveChover retorne falso. Esse fato se deve à situação climática de tempo nublado, próximo de chover, mas ainda incerto e também a intervalos de incerteza onde um mesmo valor pode representar possibilidade ou não de chuva. Tal algoritmo deve, teoricamente, ser capaz de prever cerca de 70% dos casos.

5 DESCRIÇÃO CRONOLÓGICA

Nesta seção será explicitado o roteiro do projeto determinando as atividades realizadas semanalmente no decorrer do semestre. O projeto teve início proposto para o dia 04/03/2012.

- **05/03/2012 a 12/03/2012**

  Foram discutidas as ideias das funcionalidades que seriam propostas para o projeto.

- **12/03/2012 a 19/03/2012**

  Pesquisado estado da arte e confeccionado o plano de trabalho.

- **19/03/2012 a 26/03/2012**

  Foi desenvolvida a PCB de interface de usuário, encomendados os sensores de DHT22 e BMP085 e adquirido um motor DC com caixa de redução acoplada.

- **26/03/2012 a 02/04/2012**

  Período utilizado para definir o método utilizado para a previsão de chuva, desenhar o esquemático da ponte-h – utilizando o CI L298N – e confeccionar a estrutura base do varal.

- **02/04/2012 a 09/04/2012**

  Como as atividades desta semana foram adiadas para a semana seguinte, apenas foi iniciado o desenvolvimento do conjunto pinhão/cremalheira.

- **09/04/2012 a 16/04/2012**

  Dando continuidade ao projeto eletrônico concluiu-se o projeto da ponte-h desenhando, no EagleCAD Soft, sua board.
  Na parte mecânica foi concluído o desenvolvimento do conjunto pinhão/cremalheira.
  O estudo da meteorologia teve sequência na definição dos parâmetros necessários para a previsão.

- **16/04/2012 a 23/04/2012**

Desenvolveu-se o código responsável pela leitura dos sensores de pressão e umidade.
Foi confeccionada a PCI da ponte-h e terminada a parte móvel da estrutura.

- **23/04/2012 a 30/04/2012**

  Nesta semana foi iniciado o desenvolvimento do software baseado em linguagem Processing que faria a comunicação entre o Arduino e o computador.

- **30/04/2012 a 07/05/2012**

  As atividades desta semana se resumiram ao planejamento da fixação da cremalheira e do motor, término do programa a ser utilizado na coleta de dados e confecção e impressão do sensor de chuva.

- **07/05/2012 a 14/05/2012**

  Finalizou-se a confecção do projeto mecânico fixando o carrinho de impressora. Além disto, foi definido que a melhor maneira de trabalhar com a previsão de chuva, devido à falta de tempo, era utilizando dados extraídos do banco de dados do Instituto Nacional de Meteorologia.

- **14/05/2012 a 21/05/2012**

  Definição do intervalo a ser utilizado como parâmetro de previsão do tempo.
  Confecção da placa de interface de usuário.
  Início da programação de controle de motor.

- **21/05/2012 a 28/05/2012**

  Confecção do Shield utilizado para facilitar a organização dos circuitos eletrônicos.
  Término do software de controle de motor.

- **28/05/2012 a 04/06/2012**

  Integração entre todas as partes do projeto.

- **04/06/2012 a 11/06/2012**

  Fechamento do projeto, confecção da documentação e do vídeo para a apresentação aos professores.

6 CIRCUITOS ELÉTRICOS

Nesta seção serão apresentados os esquemas, boards e fotos de todos os circuitos utilizados no projeto.

6.1 INTERFACE DE USUÁRIO

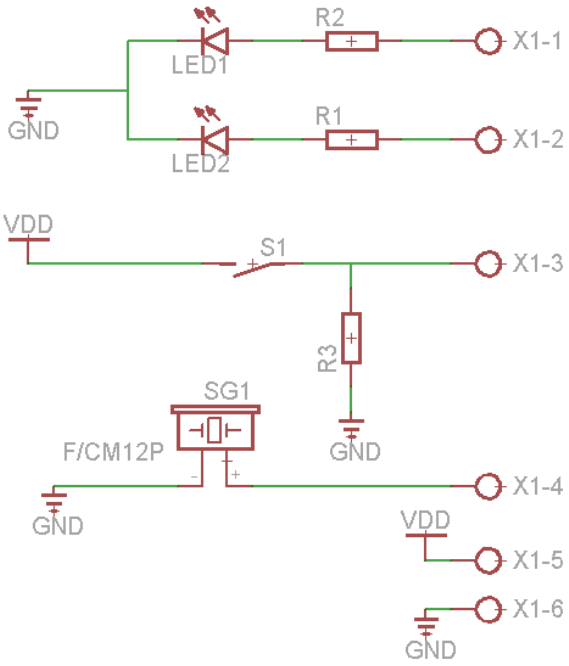

Figura 15 - Esquema da Interface de Usuário

R1 e R2 - 330Ω;

S1 - Push-button;

SG1 – Buzzer contínuo de 3V.

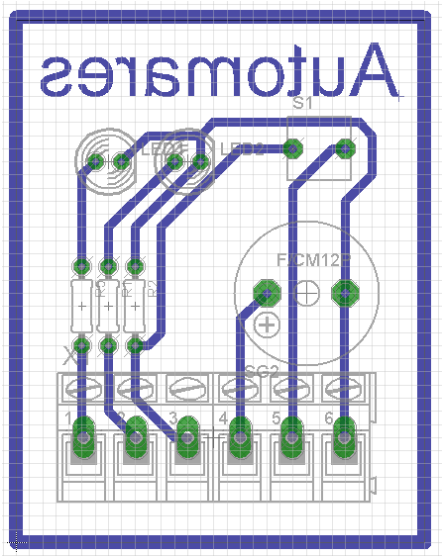

Figura 16 - Board da Interface de Usuário

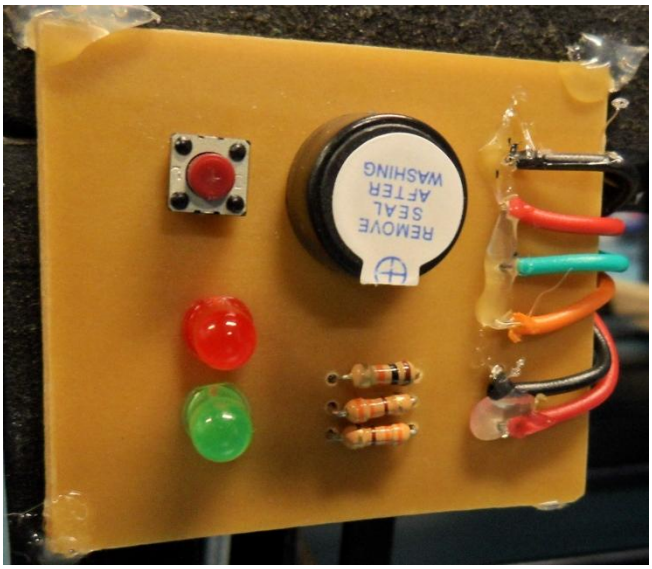

Figura 17 - Foto da Interface de Usuário

6.2 PONTE H

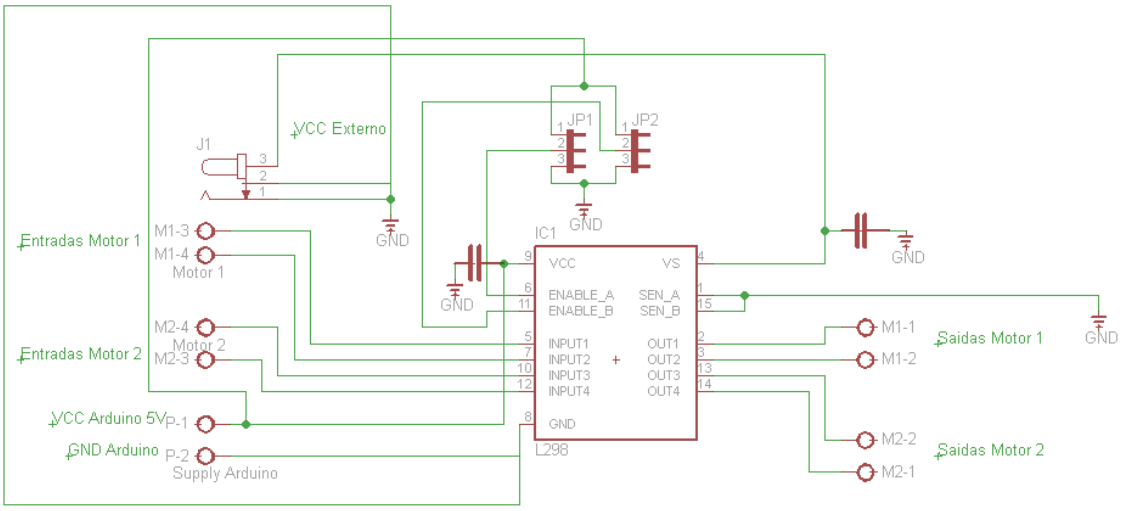

Figura 18 - Esquema Ponte H

IC1 – L298N

C1 e C2 – 100nF

J1 – Conector Jack

JP1 e JP2 – Jumpers

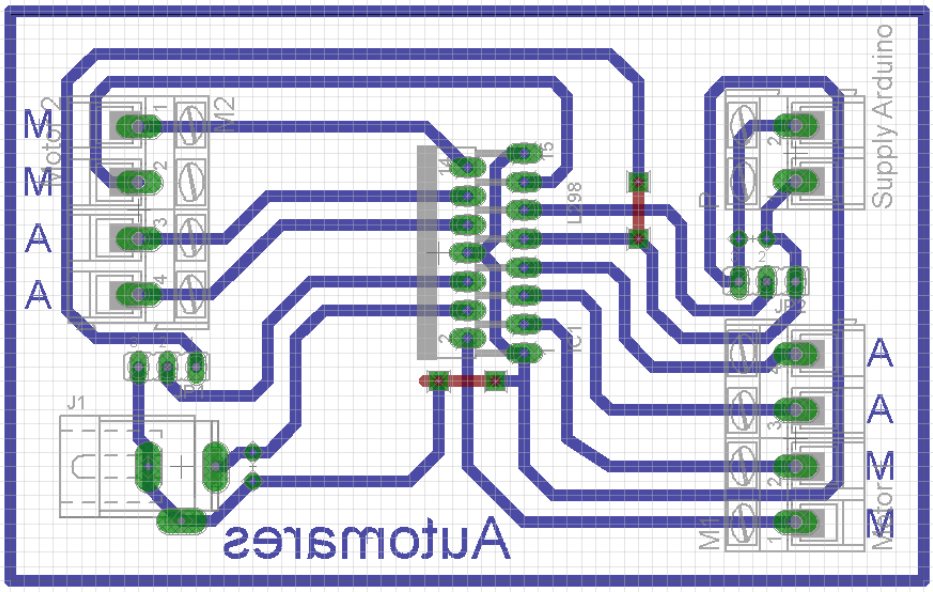

Figura 19 - Board da Ponte H

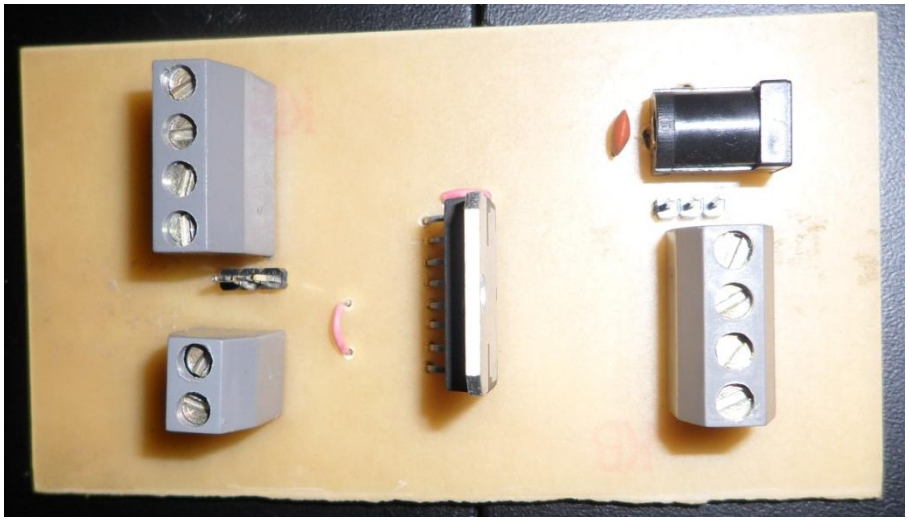

Figura 20 - Foto da Ponte H

6.3 SHIELD DE BORNES

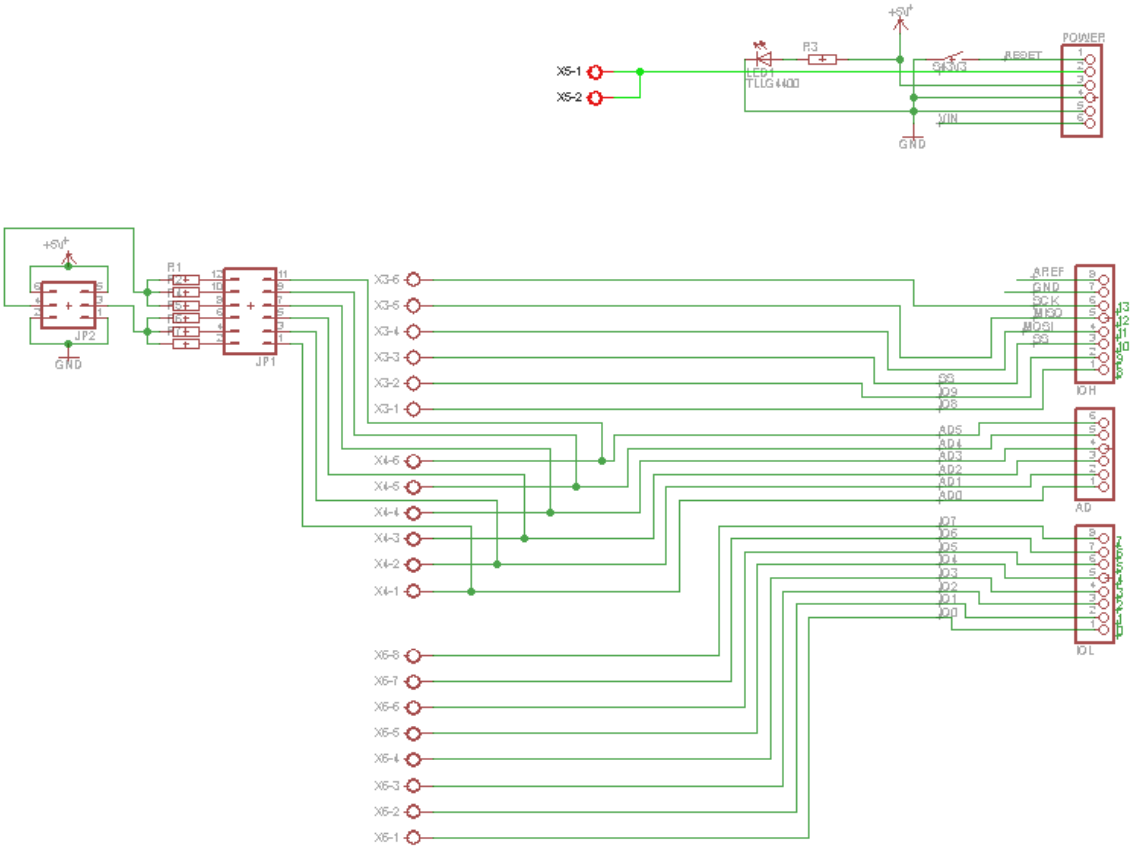

Figura 21 - Esquema do Shield de Bornes

R1, R2, R4, R5, R6 e R7 – 10KΩ

R3 - 220Ω

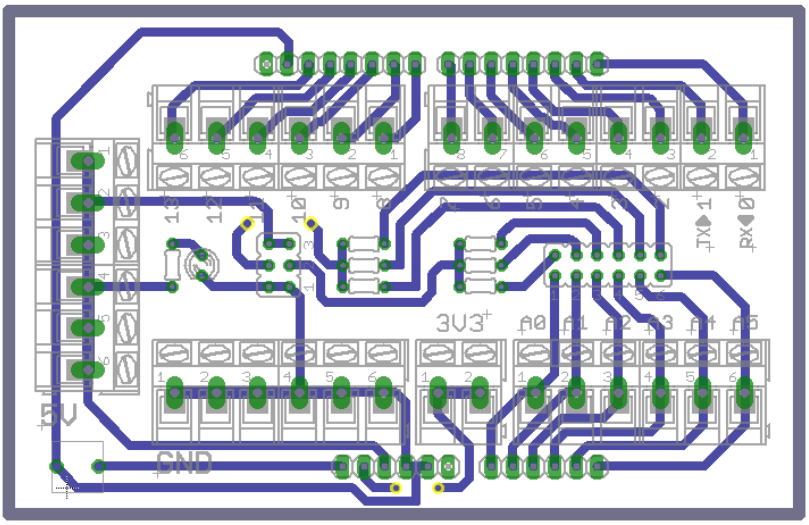

Figura 22 - Board do Shield de Bornes

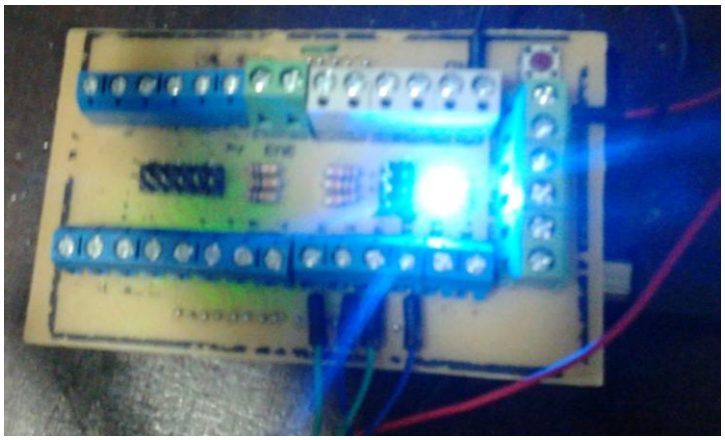

Figura 23 - Foto do Shield de Bornes

6.4 SENSOR DE CHUVA

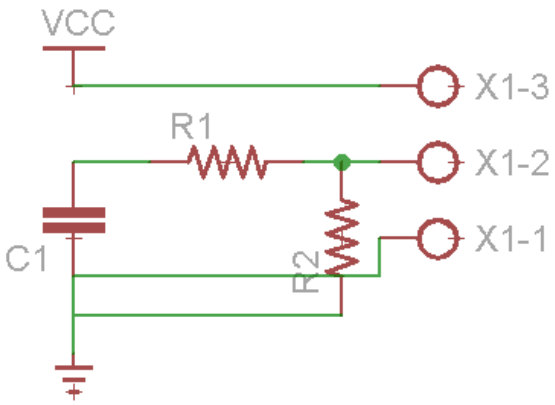

Figura 24 - Esquema do Sensor de Chuva

R1 – 100Ω

R2 – 1MΩ

C1 – 100nF

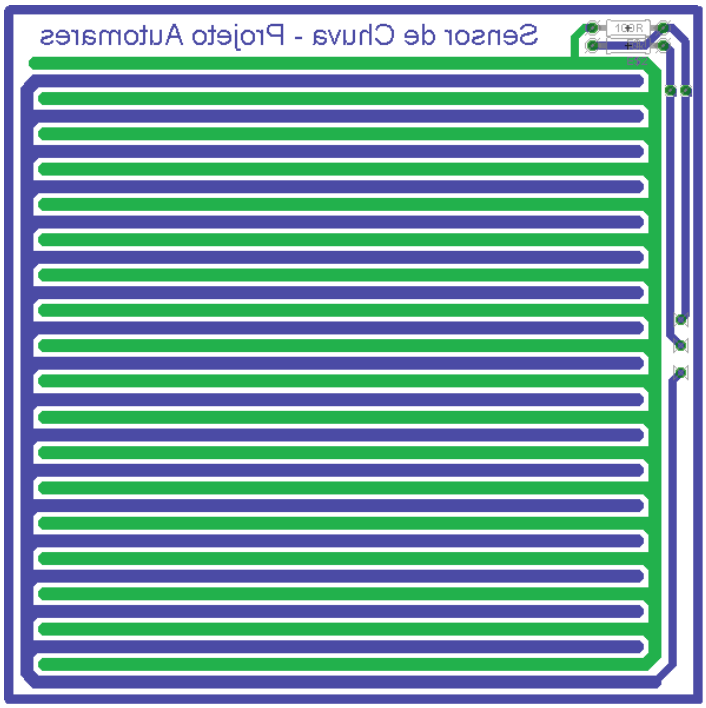

Figura 25 - Board do Sensor de Chuva

Figura 26 - Foto do Sensor de Chuva

7 CÓDIGO FONTE

7.1 MAIN

```
//-----------------------BMP085-----------------------//

#include <Wire.h>


#define BMP085_ADDRESS 0x77  // I2C address of BMP085


const unsigned char OSS = 0;  // Oversampling Setting

// Calibration values

int ac1;

int ac2;

int ac3;

unsigned int ac4;

unsigned int ac5;

unsigned int ac6;

int b1;

int b2;

int mb;

int mc;

int md;


// b5 is calculated in bmp085GetTemperature(...), this variable is also used in bmp085GetPressure(...)

// so ...Temperature(...) must be called before ...Pressure(...).

long b5;
```

```
short temperature;
long pressure;

//-----------------------DTH22-----------------------//
#include <DHT.h>
#define DHTPIN 2
#define DHTTYPE DHT22
 DHT dht(DHTPIN, DHTTYPE);

//-----------------------Motor DC-----------------------//
#define M1 11
#define M2 10
#define PinInicio 9
# define PinFim 8
long timestamp = 0;
#define DELAY 2000 //tempo de espera para tentar destravar o motor
# define DestTemp 300 // tempo que o motor será revertido para destravar

//-----------------------Interface de Usuário-----------------------//
#define LED_RED 13
#define LED_GREEN 12
#define Button 7
#define buzzer 6
boolean estendido, button=true, apito=true;
//-----------------------Sensor de chuva-----------------------//
#define SCPin 4
```

```
void setup(){
  pinMode(SCPin,INPUT);
  pinMode(M1,OUTPUT);
  pinMode(M2,OUTPUT);
  pinMode(PinInicio,INPUT);
  pinMode(PinFim,INPUT);
  pinMode(Button, INPUT);
  pinMode(LED_RED, OUTPUT);
  pinMode(LED_GREEN, OUTPUT);
  digitalWrite(LED_RED, HIGH);
  delay(500);
  if(digitalRead(PinInicio)==HIGH)
    estendido=false;
  if(digitalRead(PinFim)==HIGH)
    estendido==true;
  if(digitalRead(PinInicio)==LOW && digitalRead(PinFim)==LOW){
    estendido=false;
    Recolhe();
  }
  digitalWrite(LED_GREEN, HIGH);
  delay(500);
  Serial.begin(9600);
  dht.begin();
  Wire.begin();
  bmp085Calibracao();
}
```

```
void loop(){
  bmp085PrintTemp_e_Pres();
  PrintTemp_e_Umid();
  if(Button_Pressed()){
    if(estendido){
      button=false;
      digitalWrite(LED_RED, HIGH);
      digitalWrite(LED_GREEN, LOW);
      Recolhe();
    }
    else{
      button=true;
      if((!PodeChover() || NaoDeveChover()) && !Chovendo()){
        /*analogWrite(buzzer,255);
        delay(150);
        analogWrite(buzzer,127);
        delay(100);
        digitalWrite(buzzer,LOW);*/
        digitalWrite(LED_RED, LOW);
        digitalWrite(LED_GREEN, HIGH);
        Estende();
      }
      else{
          digitalWrite(LED_GREEN,LOW);
          delay(150);
          digitalWrite(LED_GREEN,HIGH);
          digitalWrite(LED_RED,LOW);
```

```
        analogWrite(buzzer,31);
        delay(150);
        digitalWrite(LED_GREEN,LOW);
        digitalWrite(LED_RED,HIGH);
        analogWrite(buzzer,255);
        delay(100);
        digitalWrite(buzzer,LOW);
        digitalWrite(LED_GREEN,HIGH);
        digitalWrite(LED_RED,LOW);
        delay(150);
        digitalWrite(LED_RED,HIGH);
        digitalWrite(LED_GREEN,LOW);
     }
   }
 }
  if(Chovendo() || (PodeChover() && !NaoDeveChover())){
     digitalWrite(LED_RED, HIGH);
     digitalWrite(LED_GREEN, HIGH);
     Recolhe();
  }
  else
    if((!PodeChover() || NaoDeveChover()) && button){
    digitalWrite(LED_RED, LOW);
    digitalWrite(LED_GREEN, HIGH);
    Estende();}
}
```

7.2 CONTROLE DE MOTOR

```
void Estende(){
  if(Chovendo()==false)
  while(!noFim()){
    long diferenca = millis() - timestamp;
    estendido=true;
    digitalWrite(M1,LOW);
    digitalWrite(M2,HIGH);
    if(diferenca > DELAY && !noFim()) {
      digitalWrite(M1,HIGH);
      digitalWrite(M2,LOW);
      delay(DestTemp);
      timestamp = millis();
    }
  }
    digitalWrite(M2,LOW);
}
void Recolhe(){
  long inicio = millis();
  while(!noInicio()){
    long diferenca2 = millis() - inicio;
    /*Serial.print(millis());
    Serial.print(" - ");
    Serial.print(inicio);
    Serial.print(" = ");
    Serial.println(diferenca2);*/
    estendido=false;
```

```
    digitalWrite(M1,HIGH);
    digitalWrite(M2,LOW);
    if(diferenca2 > DELAY && !noInicio()) {
      digitalWrite(M1,LOW);
      digitalWrite(M2,HIGH);
      delay(DestTemp);
      inicio = millis();
    }
  }
    digitalWrite(M1,LOW);
}
boolean noInicio(){
  if(digitalRead(PinInicio)==HIGH)
    return true;
   else
     return false;
}
boolean noFim(){
  if(digitalRead(PinFim)==HIGH)
    return true;
  else
    return false;
}
```

7.3 INTERFACE DE USUÁRIO

```
boolean Button_Pressed(){
  boolean apito=true;
  if(digitalRead(7)==HIGH){
      analogWrite(6,15);
      delay(150);
      digitalWrite(6,LOW);
      return true;
  }
  else
    return false;
}
```

7.4 LEITURA DO BMP085

```
// CALIBRAÇÃO
// Armazena todos os valores da bmp085 de calibração em variáveis globais
// Valores de calibração são necessárias para calcular a temperatura e pressão

void bmp085Calibracao(){
    ac1 = bmp085ReadInt(0xAA);
    ac2 = bmp085ReadInt(0xAC);
    ac3 = bmp085ReadInt(0xAE);
    ac4 = bmp085ReadInt(0xB0);
    ac5 = bmp085ReadInt(0xB2);
    ac6 = bmp085ReadInt(0xB4);
    b1 = bmp085ReadInt(0xB6);
    b2 = bmp085ReadInt(0xB8);
    mb = bmp085ReadInt(0xBA);
    mc = bmp085ReadInt(0xBC);
    md = bmp085ReadInt(0xBE);
}
//FUNÇÕES DE LEITURA BMP085
/****************************** BMP085 ******************************/

char bmp085Read(unsigned char address){    // Por que leitura de char?
  unsigned char data;
```

```
  Wire.beginTransmission(BMP085_ADDRESS);     // Inicia comunicação I²C com o sensor
  Wire.write(address);                        // Grava dados entre as chamadas beginTransmission e endTransmission
  Wire.endTransmission();                     // Encerra comunicação com sensor
  Wire.requestFrom(BMP085_ADDRESS, 1);        // ??
  while(!Wire.available());                   // Aguarda que os dados estejam disponíveis

  return Wire.read();
}
// Leitura de 2 bytes a partir do BMP085
// A primeira será a partir de 'address'
// O segundo byte será a partir de 'address'+1
int bmp085ReadInt(unsigned char address){
  unsigned char msb, lsb;
  Wire.beginTransmission(BMP085_ADDRESS);
  Wire.write(address);
  Wire.endTransmission();
  Wire.requestFrom(BMP085_ADDRESS, 2);
  while(Wire.available()<2);

  msb=Wire.read();
  lsb=Wire.read();

  return (int) msb<<8 | lsb;
}
```

```
unsigned int bmp085ReadUT(){ // Lê valor de temperatura não compensado | ReadUT = Lê Unconpensed Temperature

  unsigned int ut;


  // Escreve 0x2E no Registrador 0xF4

  // Este solicita uma leitura de temperatura

  Wire.beginTransmission(BMP085_ADDRESS);

  Wire.write(0xF4);

  Wire.write(0x2E);

  Wire.endTransmission();


  delay(5); //Serve para dar tempo das leituras serem feitas e pode ser substituido por uma leitura do pino EOC

        //que fica em LOW quando as leituras estão sendo feitas e em HIGH quando terminam tornando o sistema mais robusto.


  // Lê 2 bytes dos registradores 0xF6 e 0xF7

  ut = bmp085ReadInt(0xF6);

  return ut;

}


// Lê valor não compensado de Pressão

unsigned long bmp085ReadUP()

{

  unsigned char msb, lsb, xlsb;

  unsigned long up = 0;


  // Escreve 0x34 + (OSS << 6) no registrador 0xF4
```

```
  // Pede uma leitura de pressão
  Wire.beginTransmission(BMP085_ADDRESS);
  Wire.write(0xF4);
  Wire.write(0x34 + (OSS<<6));
  Wire.endTransmission();

  delay(2 + (3<<OSS));        // Lê registradores 0xF6 (MSB), 0xF7 (LSB), e 0xF8
(XLSB)
  Wire.beginTransmission(BMP085_ADDRESS);
  Wire.write(0xF6);
  Wire.endTransmission();
  Wire.requestFrom(BMP085_ADDRESS, 3);

  while(Wire.available() < 3) // Aguarda que os dados estejam disponíveis
    ;
  msb = Wire.read();
  lsb = Wire.read();
  xlsb = Wire.read();

  up = (((unsigned long) msb << 16) | ((unsigned long) lsb << 8) | (unsigned
long) xlsb) >> (8-OSS);

  return up;
}

short bmp085GetTemperature(unsigned int ut){// Calcula a temperatura dado
ut.
  long x1, x2;
```

```
  x1 = (((long)ut - (long)ac6)*(long)ac5) >> 15;
  x2 = ((long)mc << 11)/(x1 + md);
  b5 = x1 + x2;

  return ((b5 + 8)>>4);
}
// Calcula pressão dado up
//Os valores de calibração devem ser conhecidos
// b5 também é necessária para bmp085GetTemperature (...) deve ser chamado primeiro.
long bmp085GetPressure(unsigned long up)
{
  long x1, x2, x3, b3, b6, p;
  unsigned long b4, b7;

  b6 = b5 - 4000;
  // Calculate B3
  x1 = (b2 * (b6 * b6)>>12)>>11;
  x2 = (ac2 * b6)>>11;
  x3 = x1 + x2;
  b3 = (((((long)ac1)*4 + x3)<<OSS) + 2)>>2;

  // Calculate B4
  x1 = (ac3 * b6)>>13;
  x2 = (b1 * ((b6 * b6)>>12))>>16;
  x3 = ((x1 + x2) + 2)>>2;
```

```
  b4 = (ac4 * (unsigned long)(x3 + 32768))>>15;

  b7 = ((unsigned long)(up - b3) * (50000>>OSS));
  if (b7 < 0x80000000)
    p = (b7<<1)/b4;
  else
    p = (b7/b4)<<1;

  x1 = (p>>8) * (p>>8);
  x1 = (x1 * 3038)>>16;
  x2 = (-7357 * p)>>16;
  p += (x1 + x2 + 3791)>>4;

  return p;
}
void bmp085PrintTemp_e_Pres(){
  temperature = bmp085GetTemperature(bmp085ReadUT());
  pressure = bmp085GetPressure(bmp085ReadUP());
  Serial.print("Temperature: ");
  Serial.print(temperature, DEC);
  Serial.println(" *0.1 deg C");
  Serial.print("Pressure: ");
  Serial.print(pressure, DEC);
  Serial.println(" Pa");
}
short bmp085getTemperatura(){
  return bmp085GetTemperature(bmp085ReadUT());
```

```
}
long bmp085getPressao(){
  return bmp085GetPressure(bmp085ReadUP());
}
```

7.5 LEITURA DO DHT22

```
// Conectar pino 1 (da esquerda) do sensor ao +5V
// Conectar pino 2 do sensor ao DHTPIN
// Conectar pino 4 (da direita) do sensor ao GND
// Conectar um resistor de 10K do pino 2 (data) ao pino 1 (power) do sensor

void PrintTemp_e_Umid(){
  // Lê temperatura e umidade a cada 250 milisegundos
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  // Verifica se os retornos são válidos
  if (isnan(t) || isnan(h)) {
    Serial.println("Falha na comunicação com o sensor");
  } else {
    Serial.print("Umidade: ");
    Serial.print(h);
    Serial.println(" %");
    Serial.print("Temperatura: ");
    Serial.println(t);
  }
}
```

7.6 PREVISÃO DO TEMPO

```
//FUNÇÕES DE PREVISÃO DO TEMPO
//---------- Intervalos para haver chuva ----------//
#define temperatura_inferior 15.00472
#define temperatura_superior 20.77948
#define umidade_minima 80.73862
#define pressao_inferior 90962.429
#define pressao_superior 91611.171
///--------- Intervalos para ausência de chuva ---------//
#define temperatura_inferior2 15.84978
#define temperatura_superior2 25.13982
#define pressao_inferior2 91001.067
#define pressao_superior2 91617.533
#define umidade_maxima 59.1168

boolean PodeChover(){
   if ((temperatura_inferior <=
((dht.readTemperature()+(bmp085getTemperatura())*0.1)/2)) &&
(temperatura_superior >=
((dht.readTemperature()+(bmp085getTemperatura())*0.1)/2)) &&
(umidade_minima <= dht.readHumidity()) && (pressao_inferior <=
bmp085getPressao()) && (pressao_superior >= bmp085getPressao())){
     return true;
  }
   else{
     return false;
   }
```

```
 }

 boolean NaoDeveChover(){

    if(((temperatura_inferior2 <=
((dht.readTemperature()+(bmp085getTemperatura())*0.1)/2)) &&
(temperatura_superior2 >=
((dht.readTemperature()+(bmp085getTemperatura())*0.1)/2))) &&
(umidade_maxima >= dht.readHumidity()) && (pressao_inferior2 <=
bmp085getPressao()) && (pressao_superior2 >= bmp085getPressao())){

      return true;

    }

   else{

     return false;

   }

 }
```

7.7 SENSOR DE CHUVA

```
boolean Chovendo(){
  if(digitalRead(SCPin)==HIGH){
    return true;
  }
  else{
    return false;
  }
}
```

8 PROBLEMAS ENCONTRADOS

PROBLEMAS APRESENTADOS	SOLUÇÕES ENCONTRADAS
Inicialmente a ideia era criar uma função matemática para fazer a previsão de chuva, porém seria necessário um processamento muito superior ao que o Arduino poderia oferecer.	Definir um intervalo de comparação relacionando pressão, temperatura e umidade.
O shield do sensor de pressão e temperatura BMP085 queimou devido a uma ligação feita de forma inadequada da alimentação.	Adquirir um novo shield.
Mal funcionamento da ponte-h. Uma nova ponte h foi refeita porém devido a um possível erro de projeto ou má utilização não obteve-se sucesso.	Utilizar uma ponte-h previamente construída por um dos integrantes.
A ideia de utilizar um conjunto cremalheira/pinhão teve que ser abandonada devido ao alto custo de fabricação dos mesmos.	Adaptar um carrinho de impressora para conduzir a estrutura móvel.
Ao tentar utilizar a técnica de PWM para controlar o motor foi observado um ruído sonoro de alta frequência.	Utilizar apenas comandos digitais orientados por fins de curso mecânicos colocados nas extremidades da estrutura.

9 CONCLUSÃO

Aplicando os conhecimentos obtidos até o presente momento foi possível alcançar os objetivos traçados para o projeto. A principal proposta do projeto, expressa pela disciplina de RPE, foi percebida de maneira extremamente clara, visto que diversas vezes o grupo se deparou com problemas diferentes dos propostos em sala de aula e, ao buscar soluções, compreendeu os problemas cotidianos de um profissional da área de engenharia.

No início do projeto houve bastante dificuldade com relação à divisão e execução de tarefas em áreas com pouco conhecimento ou fora do escopo proposto pelo curso. Ao concluir o projeto e verificar que seu funcionamento atendeu as expectativas foi gratificante observar a superação das adversidades encontradas inicialmente.

10 REFERÊNCIAS

BMP085 Quickstart Guide. Disponível em: http://www.sparkfun.com/tutorials/253. Acesso: 21 abril. 2012.

DHT11/DHT21/DHT22 etc. Temperature & Humidity sensors. Disponível em: http://www.ladyada.net/learn/sensors/dht.html. Acesso: 21 abril. 2012.

Observando Pressão e Temperatura. Disponível em: http://www.popa.com.br/meteorologia/pressao&temperatura.htm. Acesso: 29 março. 2012.

Processing. Disponível em: http://processing.org/. Acesso: 27 abril. 2012.

INMET. Disponível em: http://www.inmet.gov.br/sim/sonabra/dspDadosCodigo.php?ODM4NDI=. Acesso em: 10 maio. 2012.

Ler a Serial e salvar dados txt do Arduino. Disponível em: http://fisicacomputacaopucsp.blogspot.com.br/2011/04/ler-serial-e-salvar-dados-txt-do.html. Acesso: 10 maio. 2012.

Leitura de temperatura com arduino. Disponível em: http://blogdoje.com.br/2007/07/08/leitura-de-temperatura-com-arduino/. Acesso: 10 maio. 2012.

11 GLOSSÁRIO

L298N/L293D - CI de ponte-h dupla utilizado para controlar até dois motores.

Buzzer - Beeper. Dispositivo de sinalização sonora comumento utilizado em eletrônica.

LED - Diodo emissor de luz, ou do inglês Light Emitting Diode. Utilizado em eletrônica para sinalização visual.

Push-button - Botão. Dispositivo utilizado para cortar ou permitir a passagem de corrente elétrica.

Circuito - Circuito elétrico. Ligação de elementos elétricos de tal forma que forneça pelo menos um caminho fechado para a corrente elétrica.

Board - Nome dado ao desenho a ser impresso na na placa de fenolite.

Resistor de pull Down - Resistor utilizado em circuitos elétricos para garantir o estado lógico "baixo".

Capacitor - Dispositivo utilizado para armazenar energia temporariamente.

VCC - Voltagem Corrente Contínua. Sigla utilizada para denominar o ponto de maior potencial do circuito.

GND - Ground (do inglês). Terra. Sigla utilizada para denominar o ponto de menor potencial do circuito.

Fim-de-curso - Sensor de toque. Dispositivo mecânico responsável por indicar ao controlador a interrupção do movimento.

Arduino - Plataforma de prototipagem eletrônica de hardware livre, projetada com um microcontrolador ATmega328. É programado com uma linguagem de programação própria baseada na linguagem C.

PCI/PCB - Placa de circuito impresso ou, do inglês, Printed circuit board. Placa contendo o circuito impresso em uma superfície de cobre.

EagleCAD Soft - Software utilizado para o desenho de circuitos a serem impressos.

CI - Circuito Integrado. Circuito eletrônico miniaturizado.

Shield - Placas que podem ser plugadas no topo do arduino, extendendo suas funções.

Ponte-h - Circuito utilizado para controle de motores DC.

Motor DC - Motor alimentado com corrente contínua.

Linguagem de Programação - Método padronizado para comunicar isntruções para um computador. Conjunto de regras sintáticas e semânticas usadas para definir um programa de computador.

Função - Desvio de código que agrupa comandos que são utilizados diversas vezes durante o programa