Projeto de auxílio à integração da eletrônica à engenharia agrícola

Projeto Final

Vitor Jacinto Sulzbach
Universidade de Brasília - Faculdade do gama
UnB - FGA
Gama, Distrito Federal, Brasil
vjsulzbach@gmail.com

Diego Galdino Mendonça
Universidade de Brasília - Faculdade Gama
Unb - FGA
Gama, Distrito Federal, Brasil
diegaozims@gmail.com

Keywords—meteorology; agriculture; electronic

I. RESUMO

Integração de sensores voltados para a agricultura de precisão com uma plataforma online de IoT utilizando as placas MSP430G2553 e a NodeMCU ESP12e.

II. INTRODUÇÃO

Um dos grandes desafios atuais da engenharia agrícola é a crescente introdução da eletrônica por meio da agricultura de precisão, que exige o monitoramento constante com auxílio de imagens de satélites e drones, além de sistema de geolocalização ultra preciso para o maquinário e diversos mapas para controle de nutrição, praga, topologia, entre outros.

A maioria das fazendas não conta com aparato para realizar medições de características climáticas e agronômicas importantes que afetam sua produtividade. Isto é influência direta da falta de domínio sobre projeto de sistemas eletrônicos que grande parte dos profissionais da área possuem.

Para melhorar o acesso do campo aos benefícios da agricultura de precisão se faz necessário a existência de profissionais com maior conhecimento em eletrônica, porém respeitando o fato de que o verdadeiro foco desta área é a agricultura e não o estudo de sistemas eletrônicos surge a ideia de simplificar os mesmos trazendo conhecimento filtrado para não sobrecarregar os interessados com conhecimentos desnecessários para o propósito final, que é o aumento da produção agrícola, então se propõe realizar um estudo sobre sensores de fácil acesso e baixo custo para futuros profissionais da área.

III. DESENVOLVIMENTO

Visto que se trata de um projeto voltado para auxiliar majoritariamente estudantes é necessário utilizar sensores de baixo custo e fácil acesso, contudo também devem ter precisão e operação aceitáveis. Através de um levantamento de preços e disponibilidades em diversos sites de compra na internet se obteve a seguinte lista sensores que atendem os requisitos.

Sensor	**Preço**
Módulo + Sensor de chuva	R$ 6,90
Módulo + sensor de umidade do solo	R$ 6,15
TIL78 3mm	R$ 0,68
Total sem frete	R$ 13,68

Tabela 1 - Preço dos Sensores

1. Irradiação Solar

Toda cultura depende da irradiação para crescer, afinal o metabolismo inteiro da planta depende da fotossíntese, que é uma conversão da radiação solar para energia eletroquímica. Para medir tão grandeza foi utilizado um TIL78 de 3 milímetros, que é um transistor capaz de regular a corrente Ice através da luminosidade em sua base.

O fototransistor é alterado através da geração de pares eletro-lacunas no silício que se formam pela energia dissipada de fótons em sua superfície. Através da polarização de uma resistência é possível obter um valor na placa MSP430 que é proporcional à irradiação. A imagem abaixo demonstra o funcionamento do circuito. A luz é recebida pelo transistor que regula a corrente no potenciômetro a ser polarizado, o capacitor serve para impedir mudanças muito abruptas de tensão, além de filtrar ruídos de alta frequência, a tensão é lida em V_{out}. A alteração do valor de resistência do potenciômetro altera diretamente o valor da tensão e por consequência a sensibilidade e valores medidos pelo MSP.

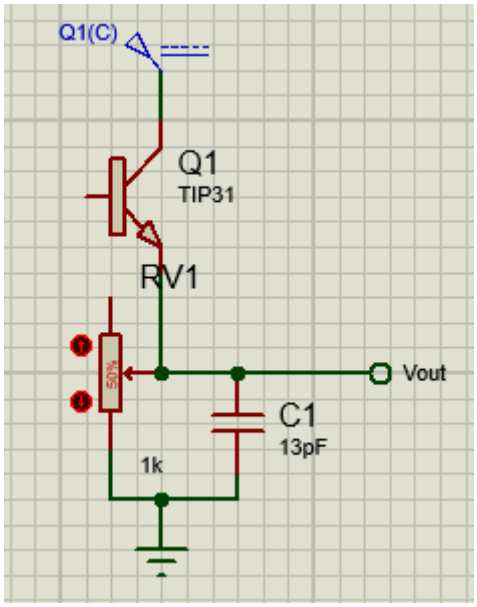


Figura 1 – Coleta da irradiação

2. Sensor binário de chuva

O sensor de chuva é uma placa com trilhas de metal que estão presentes por toda superfície, mas não se encontram. O circuito fica aberto até algo o fechar, neste caso se pressupõe que seja a chuva. O sensor também acompanha um módulo comparador que retira a necessidade de uma conversão A/D combinada a lógica de comparação.

É notável que o sensor possua leitura analógica sendo que o propósito é binário, ou seja, o propósito é saber se está ou não chovendo. Isto se deve ao fato de que a impedância do sensor varia conforme sua superfície é coberta com água, está cobertura depende da inclinação do sensor e da quantidade de chuva caindo no momento, é de se esperar que a integral deste valor instantâneo seja proporcional à precipitação no período da integral, porém este cálculo só vale para um sistema teórico pois o sistema real possui grande influência da aleatoriedade tanto em relação à disposição das gotas na placa quanto à quantidade de íons e PH da água da chuva que cai no momento, estes fatores impedem uma leitura aceitável o que leva a busca de outros métodos para leitura da precipitação.

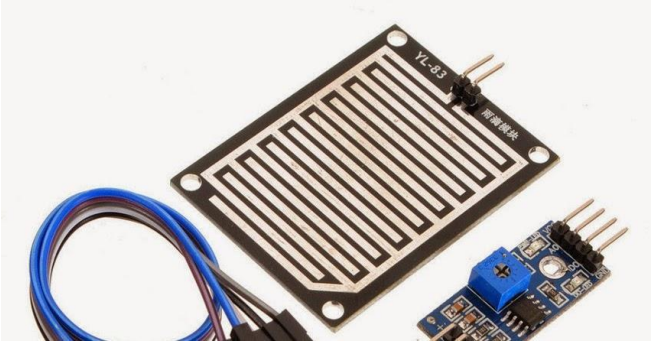


Figura 2 – Sensor de Chuva

3. Umidade do Solo

A umidade do solo é de extrema importância para determinar a absorção de água da cultura naquele local. O mesmo trabalha medindo a impedância do solo que está entre suas duas pontas, acompanha um módulo que facilita sua leitura, mas pode ser lido com um circuito idêntico ao do fototransistor, porém trabalhando com o princípio de divisão de tensão.

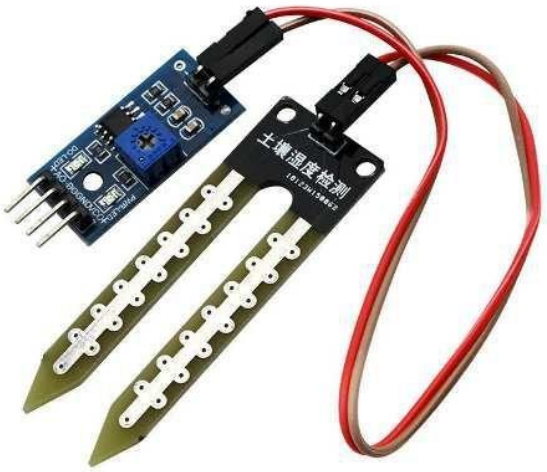


Figura 3 – Sensor de impedância do solo

Uma característica importante deste sensor é que o mesmo não mede diretamente a umidade do solo e sim a impedância, ou seja, o valor mensurado depende de outros fatores como a quantidade de sais ionizados no solo e nutrientes em geral, além da própria composição físico-química do solo, portanto se faz necessário calibrar o dispositivo no local inserido

periodicamente devido a alteração que os ciclos de cultivo causam no solo.

4. Montagem

Estes sensores são lidos pela MSP, que une a o sinal em um array de chars e envia o mesmo para a placa NodeMCU via comunicação UART, a NodeMCU recebe estes valores, os decompõe e envia para a nuvem do thingspeak a cada 45 segundos, como mostra o diagrama de blocos abaixo.

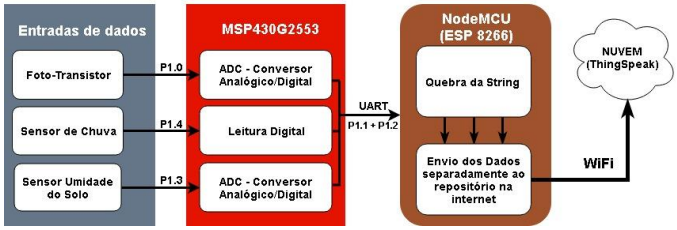


Figura 4 - Diagrama de blocos do circuito(Obs: imagem ampliada no Anexo 1)

5. Lista de materiais(Bill of Materials)

Material	Quantidade
Módulo + Sensor de chuva	1
Módulo + sensor de umidade do solo	1
TIL78 3mm	1
Protoboard	1
MSP430G2553	1
NodeMCU (ESP 8266)	1
Jumper Macho-Fêmea	Pacote 40
Jumper Fêmea-Fêmea	Pacote 40
Jumper Macho-Macho	Pacote 40
Potenciômetro 100k Ohms	1

Tabela 2 - Lista de materiais utilizados na execução do projeto

6. Hardware

O hardware utilizado pode ser considerado bem simples, utilizando um circuito de polarização, como explicado no tópico sobre irradiação solar e componentes digitais interconectados.

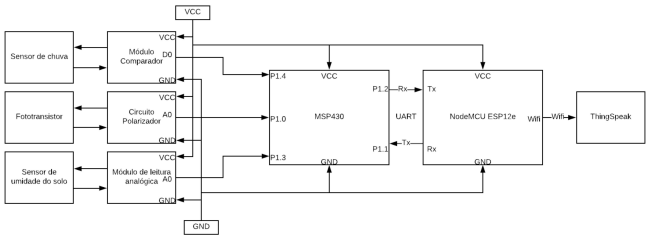


Figura 5 - Hardware

7. Software

A Placa MSP430 foi programada através do software code composer utilizando linguagem C. Seu main é dividido em diversas funções como mostra a Figura 6.

```
8 int main(void)
9 {
10     MSP_Set();
11     ADC_init();
12     UART_Set();
13     while(1) {
14         int umidade_solo = Leitura_umidade();
15         int luminosidade = Leitura_lum();
16         int chuva = Leitura_chuva();
17         UART_Send(umidade_solo, luminosidade, chuva);
18         __delay_cycles(2500000);
19     }
20 }
```

Figura 6 - Main MSP

Primeiro a MSP tem seu clock fixado em um megaHertz, o whatchdog timer é desligado e as entradas são setadas, tudo através da função MSP_Set().

```
37 void MSP_Set() {
38     WDTCTL = WDTPW + WDTHOLD;
39     BCSCTL1 = CALBC1_1MHZ;
40     DCOCTL = CALDCO_1MHZ;
41     P1DIR &= ~BIT4;
42     P1DIR &= ~BIT3;
43     P1DIR &= ~BIT0;
44 }
45
```

Figura 7 - MSP_Set

Depois o ADC e a UART são configuradas conforme a Figura 8 mostra.

```
void ADC_init() {
    ADC10CTL0 = SREF_0 + ADC10SHT_0 + ADC10ON;
}

void UART_Set() {
    P1DIR |= BIT2;
    P1DIR &= ~(BIT1);
    P1SEL |= (BIT1 + BIT2);
    P1SEL2 |= (BIT1 + BIT2) ;
    UCA0CTL1 |= UCSSEL_2;
    UCA0BR0 = 104;
    UCA0BR1 = 0;
    UCA0MCTL = UCBRS0;
    UCA0CTL1 &= ~UCSWRST;
}
```

Figura 8 - UART e MSP set

Depois a placa entra em loop infinito, sempre lendo os valores dos sensores e enviando os mesmos juntos em uma string via UART.

As funções da figura 9 mostram as diferentes configurações para leitura da porta P1.3 e P1.0.

```
void ADC_set3() {
    ADC10CTL0 &= ~(ENC + ADC10SC);
    ADC10AE0 = BIT3;
    ADC10CTL1 = INCH_3 + ADC10DIV_0 + ADC10SSEL_3 + CONSEQ_0 + SHS_0;
}
void ADC_set0() {
    ADC10CTL0 &= ~(ENC + ADC10SC);
    ADC10AE0 = BIT0;
    ADC10CTL1 = INCH_0 + ADC10DIV_0 + ADC10SSEL_3 + CONSEQ_0 + SHS_0;
```

Figura 9 - ADC Set para P1.3 e P1.4

A figura 10 mostra a função que lê e retorna o valor do conversor A/D.

```
int ADC_read() {
    ADC10CTL0 |= ENC + ADC10SC;
    while((ADC10CTL0 & ADC10IFG)==0);
    __delay_cycles(10000);
    int a = ADC10MEM;
    return a;
}
```

Figura 10 - Leitura do conversor A/D

A figura 11 exibe como é feita a leitura de umidade, luminosidade e de chuva.

```
int Leitura_umidade() {
    ADC_set3();
    int a = ADC_read();
    return a;
}

int Leitura_lum() {
    ADC_set0();
    int a = ADC_read();
    return a;
}

int Leitura_chuva() {
    if((P1IN & BIT4) == BIT4) {
        return 1;
    } else {
        return 0;
    }
```

Figura 11 - Leitura

A figura 12 mostra como é feita a comunicação UART.

```
void UART_Send(int a, int b, int c) {
    nr_of_chars = sprintf(buffer, "%d %d %d \n", a, b, c);
    Send(buffer);
}

void Send(char* tx_data) {
    unsigned int i=0;
    while(tx_data[i])
    {
        while ((UCA0STAT & UCBUSY));
        UCA0TXBUF = tx_data[i];
        i++;
    }
```

Figura 12 - Funções para comunicação UART

A NodeMCU foi programada usando a variação de C da empresa Arduino. Inicialmente na fase de setup inicia-se a comunicação Wi-fi, serial e com a thingspeak, além de zerar as variáveis globais.

```
void setup()
{
    Serial.begin(115200);
    FazConexaoWiFi();
    ThingSpeak.begin(client);
    Serial.println("Planta IoT com ESP8266 NodeMCU");
    i=0;
    f=0;
}
```

Figura 13 - Setup

Já em loop o programa lê a comunicação UART, quebra a String em valores inteiros e mostra no monitor serial. Este processo é executado 15 vezes, uma vez por segundo, até que

se envie um dos dados, um de cada vez, completando um ciclo completo.

```
Serial.println("-----------------------------------------------");
Serial.println();
int u = 16-i;
Serial.print("Faltam ");
Serial.print(u);
Serial.println(" ciclos para enviar novamente");
m = Serial.readString();
String n,j,k = "";
n = getValue(m,' ',0);
j = getValue(m,' ',1);
k = getValue(m,' ',2);
Serial.print("Umidade: ");
Serial.println(n);
Serial.print("Luminosidade: ");
Serial.println(j);
Serial.print("Chovendo: ");
Serial.println(k);
if(i>15){
  Serial.println();
  switch (f) {
    case 0:
    f++;
    Serial.println("Enviando Umidade...");
    ThingSpeak.writeField(myChannelNumber, 1, n.toInt(), ChaveEscritaThingSpeak1);
    break;

    case 1:
    Serial.println("Eviando Luminosidade...");
    f++;
    ThingSpeak.writeField(myChannelNumber, 2, j.toInt(), ChaveEscritaThingSpeak1);
    break;

    case 2:
    Serial.println("Enviando Sensor de Chuva...");
    f=0;
    ThingSpeak.writeField(myChannelNumber, 3, k.toInt(), ChaveEscritaThingSpeak1);
    break;

    default:
    break;
  }
  i=0;
}
i++;
```

Figura 14 - Void Loop

IV. RESULTADOS

Foi possível realizar a comunicação com a API ThingSpeak, atualizando cada um dos sensores de 45 em 45 segundos. Alguns valores iguais a 0 contaminaram o gráfico obtido devido à compilação e carregamento dos códigos exigirem a desconexão da comunicação UART, a NodeMCU envia 0 nesta situação, que é corrigida logo em seguida com o retorno do UART. Uma vez compilada e carregada as placas não precisaram passar por este processo novamente no caso de serem instaladas em campo, impossibilitando o aparecimento destes contaminantes.

Outra possível melhoria é a conversão do sensor de impedância do solo para Ohms no espaço de ThingSpeak, que facilitaria a utilização do projeto. Também uma boa prática seria enviar os dados simultâneamente para a nuvem para que a fase da amostragem dos sinais estejam em concordância, o que facilita o relacionamento dos dados.

Seria uma boa prática temporizar a MSP, utilizando um timer e uma interrupção para deixar o controlador em modo de baixo consumo enquanto não está executando nenhuma tarefa.

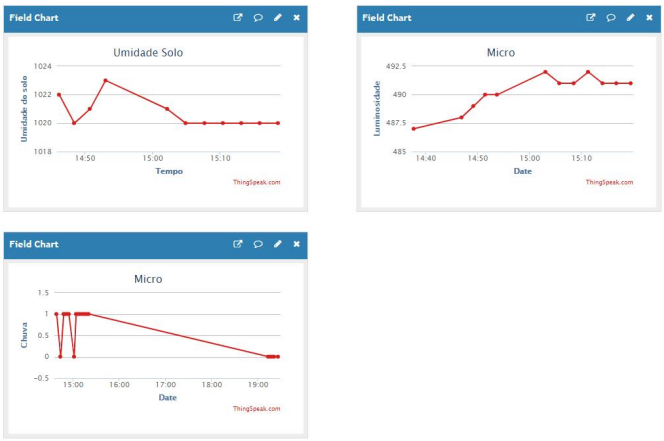


Figura 15 - Gráficos do ThingSpeak

V. CONCLUSÃO

Com o objetivo de auxiliar o processo de aquisição de dados para profissionais da área agrícola o projeto conseguiu capturar grandezas de umidade do solo e irradiação solar além de detectar a chuva no local de instalação, também foi implementado uma comunicação Wi-fi que transmite estes dados para a API ThingSpeak.

VI. REVISÃO BIBLIOGRÁFICA

[1] BOYLESTAD, Robert. Introdução à Análise de Circuitos. 10ª. São Paulo : Pearson Prentice Hall, 2004.

[2] INMETRO, "Instrumentos de Medição". Disponível em: <http://www.inmetro.gov.br/consumidor/instrumentosMedicao.asp> Acesso em: 01 de agosto de 2018.

[3] AGSOLV, "Boas Práticas para o Funcionamento de Sensores Meteorológicos". Disponível em: <https://www.agsolve.com.br/dicas-e-solucoes/10416/boas-praticas-para-o-funcionamento-de-sensores-meteorologicos> Acesso em: 01 de agosto de 2018.

[4] AGROSMART, "Estação meteorológica: como funciona e sua importância na agricultura". Disponível em:<https://agrosmart.com.br/blog/irrigacao/estacao-meteorologica-funciona-importancia-agricultura/> Acesso em: 02 de agosto de 2018.

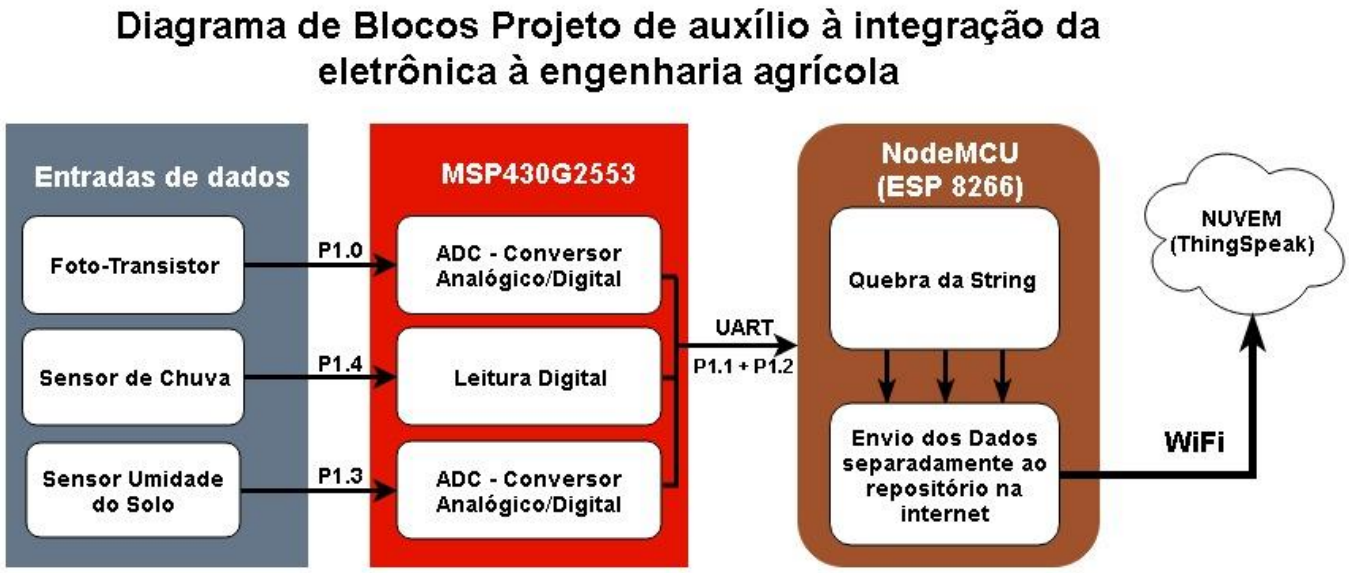
[5] EMBARCADOS, "Estação meteorológica com Arduino". Disponível em: <https://www.embarcados.com.br/estacao-meteorologica-com-arduino/> Acesso em: 02 de agosto de 2018.

[7] Mundo Clima – Equipamentos de medição climática. Disponível em: <https://www.mundoclima.com.br/estacoes-meteorologicas/po

rtateis/estacao-meteorologica-portatil-kestrel-3000/> Acesso em: 01 de agosto de 2018.

[8] Mundo Clima – Equipamentos de medição climática. Disponível em: <https://www.mundoclima.com.br/estacoes-meteorologicas/portateis/estacao-meteorologica-kestrel-5200-professional/> Acesso em: 01 de agosto de 2018.

Anexo 1



Anexo 2

Código da ESP

```
#include <ESP8266WiFi.h> //essa biblioteca já vem com a IDE. Portanto, não é preciso baixar nenhuma biblioteca adicional
#include <ThingSpeak.h>

//defines
#define SSID_REDE     "DESKTOP-QT29U8E 3407"  //coloque aqui o nome da rede que se deseja conectar
#define SENHA_REDE    "832$2d8D"  //coloque aqui a senha da rede que se deseja conectar
#define INTERVALO_ENVIO_THINGSPEAK  1200  //intervalo entre envios de dados ao ThingSpeak (em ms)

//constantes e variáveis globais
```

```
const char * ChaveEscritaThingSpeak1 = "K3PYIKR2DDFR514W";
unsigned long myChannelNumber = 643056;
int i, f;
WiFiClient client;

void FazConexaoWiFi(void)
{
   client.stop();
   Serial.println("Conectando-se à rede WiFi...");
   Serial.println();
   delay(1000);
   WiFi.begin(SSID_REDE, SENHA_REDE);

   while (WiFi.status() != WL_CONNECTED)
   {
      delay(500);
      Serial.print(".");
   }

   Serial.println("");
   Serial.println("WiFi connectado com sucesso!");
   Serial.println("IP obtido: ");
   Serial.println(WiFi.localIP());

   delay(1000);
}

void setup()
{
   Serial.begin(115200);
   FazConexaoWiFi();
   ThingSpeak.begin(client);
   Serial.println("Planta IoT com ESP8266 NodeMCU");
   i=0;
   f=0;
}
```

```
void loop()
{
  String m; Serial.println("-----------------------------------------");
  Serial.println();
  int u = 16-i;
  Serial.print("Faltam ");
  Serial.print(u);
  Serial.println(" ciclos para enviar novamente");
  m = Serial.readString();
  String n,j,k = "";
  n = getValue(m,' ',0);
  j = getValue(m,' ',1);
  k = getValue(m,' ',2);
  Serial.print("Umidade: ");
  Serial.println(n);
  Serial.print("Luminosidade: ");
  Serial.println(j);
  Serial.print("Chovendo: ");
  Serial.println(k);
  if(i>15){
    Serial.println();
    switch (f) {
      case 0:
      f++;
      Serial.println("Enviando Umidade...");
      ThingSpeak.writeField(myChannelNumber, 1, n.toInt(), ChaveEscritaThingSpeak1);
      break;

      case 1:
      Serial.println("Eviando Luminosidade...");
      f++;
      ThingSpeak.writeField(myChannelNumber, 2, j.toInt(), ChaveEscritaThingSpeak1);
      break;

      case 2:
      Serial.println("Enviando Sensor de Chuva...");
      f=0;
```

```
    ThingSpeak.writeField(myChannelNumber, 3, k.toInt(), ChaveEscritaThingSpeak1);
    break;

    default:
    break;
   }
   i=0;
  }
  i++;
  Serial.println();
  Serial.println("-----------------------------------------");
  delay(1000);
}

String getValue(String data, char separator, int index)
{
  int found = 0;
  int strIndex[] = {0, -1};
  int maxIndex = data.length()-1;

  for(int i=0; i<=maxIndex && found<=index; i++){
   if(data.charAt(i)==separator || i==maxIndex){
      found++;
      strIndex[0] = strIndex[1]+1;
      strIndex[1] = (i == maxIndex) ? i+1 : i;
   }
  }

  return found>index ? data.substring(strIndex[0], strIndex[1]) : "";
}
```

Anexo 2

Código da MSP

```
#include <msp430g2553.h>
#include <stdio.h>
#include <stdint.h>
char buffer[50];
uint8_t nr_of_chars;


int main(void)
{
    MSP_Set();
    ADC_init();
    UART_Set();
    while(1) {
       int umidade_solo = Leitura_umidade();
       int luminosidade = Leitura_lum();
       int chuva = Leitura_chuva();
       UART_Send(umidade_solo, luminosidade, chuva);
       _delay_cycles(2500000);
    }
}

void UART_Send(int a, int b, int c) {
  nr_of_chars = sprintf(buffer, "%d %d %d \n", a, b, c);
  Send(buffer);
}

void Send(char* tx_data) {
  unsigned int i=0;
  while(tx_data[i])
  {
     while ((UCA0STAT & UCBUSY));
```

```
        UCA0TXBUF = tx_data[i];
        i++;
    }
}

void MSP_Set() {
    WDTCTL = WDTPW + WDTHOLD;
    BCSCTL1 = CALBC1_1MHZ;
    DCOCTL = CALDCO_1MHZ;
    P1DIR &= ~BIT4;
    P1DIR &= ~BIT3;
    P1DIR &= ~BIT0;
}

int Leitura_umidade() {
    ADC_set3();
    int a = ADC_read();
    return a;
}

int Leitura_lum() {
    ADC_set0();
    int a = ADC_read();
    return a;
}

int Leitura_chuva() {
    if((P1IN & BIT4) == BIT4) {
        return 1;
    } else {
        return 0;
    }
}

void ADC_init() {
    ADC10CTL0 = SREF_0 + ADC10SHT_0 + ADC10ON;
}
```

```
void UART_Set() {
  P1DIR |= BIT2;
  P1DIR &= ~(BIT1);
  P1SEL |= (BIT1 + BIT2);
  P1SEL2 |= (BIT1 + BIT2) ;
  UCA0CTL1 |= UCSSEL_2;
  UCA0BR0 = 104;
  UCA0BR1 = 0;
  UCA0MCTL = UCBRS0;
  UCA0CTL1 &= ~UCSWRST;
}

void ADC_set3() {
  ADC10CTL0 &= ~(ENC + ADC10SC);
  ADC10AE0 = BIT3;
  ADC10CTL1 = INCH_3 + ADC10DIV_0 + ADC10SSEL_3 + CONSEQ_0 + SHS_0;
}
void ADC_set0() {
  ADC10CTL0 &= ~(ENC + ADC10SC);
  ADC10AE0 = BIT0;
  ADC10CTL1 = INCH_0 + ADC10DIV_0 + ADC10SSEL_3 + CONSEQ_0 + SHS_0;
}

int ADC_read() {
  ADC10CTL0 |= ENC + ADC10SC;
  while((ADC10CTL0 & ADC10IFG)==0);
  _delay_cycles(10000);
  int a = ADC10MEM;
  return a;
}
```

Anexo 2

Hardware