Projeto sensor de umidade

Microcontroladores e microprocessadores

Integrantes do grupo:

João Ricardo de Lima Bonifacio RA: 0791611015

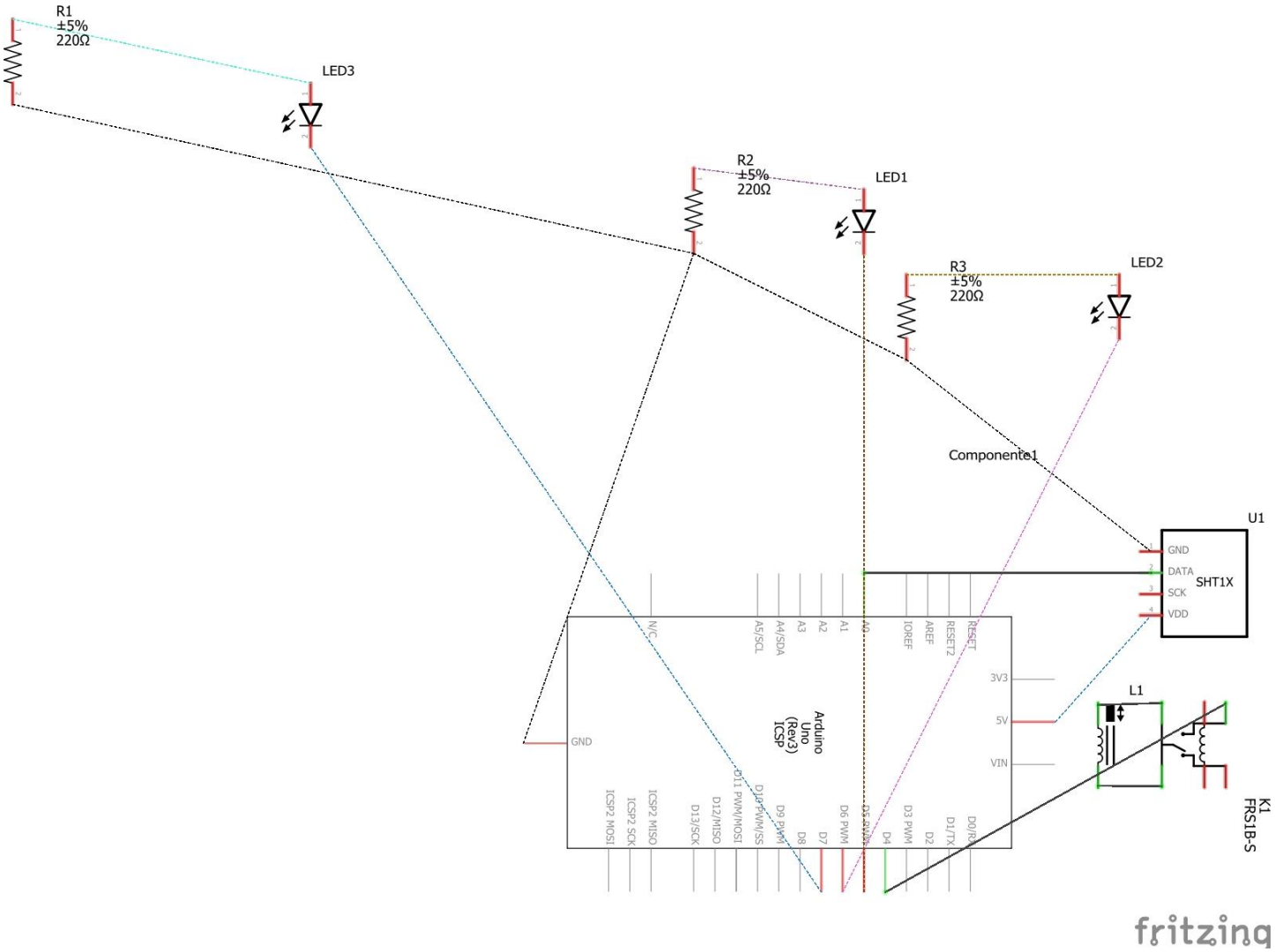
Juan Valdestilhas Daminello RA: 0791711016

Thays de Andrade Zanin RA: 0791521034

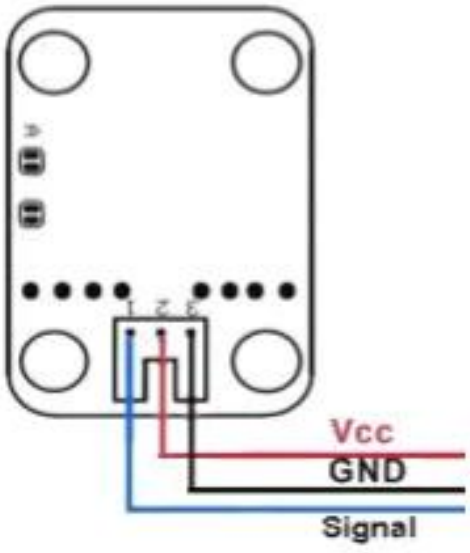
Como funciona

O circuito se inicia com o sensor enviando um sinal analógico, sendo convertido para digital dentro da placa e sinalizando através dos LEDs o nível de umidade do solo; Então caso o solo esteja muito seco ou em um nível médio, será acionada a válvula solenoide do circuito, liberando agua e umidificando o solo até que ele esteja novamente com um nível aceitável de umidade.

Circuito



Data sheet dos sensores



YL-38	
Pino	**Função**
Vcc - 1	VCC
Gnd - 2	Gnd
D0 - 3	Digital
A0 - 4	Analógico

Lista de componentes

- 3x LEDs;

- 3x Resistores 220Ω;

- Módulo rele 5V 10A;

- Sensor de umidade do solo;

- Válvula solenoide;

- Arduino Uno

- Protoboard;

Programa

- Sistema de irrigação

```
#include <avr/io.h>

#include <util/delay.h>

#include “minhasCom.h”

#include “minhasMacros.h”

#include “minhaAdc.h”

#include “Usart0.h”

#define pino_sinal_analogico A0 << PC5

#define pino_led_vermelho 5 << PB4

#define pino_led_amarelo 6 << PB3

#define pino_led_verde 7 << PB2

#define pino_bomba 9 << PB1

#define pino_sinal_analogico A0 MUX0

#define led_vermelho 5 MUX1

#define led_amarelo 6 MUX2

#define led_verde 7 MUX3

#define bomba 9 MUX4

int valor_analogico;

void setup()
```

```
{
 initUsart0();
 setBit(DDRB, DDD5);
 clearBit(PORTB, PC5);
 setBit(DDRB, DDD5);
 clearBit(PORTB, PB4);
 setBit(DDRB, DDD5);
 clearBit(PORTB, PB3);
 setBit(DDRB, DDD5);
 clearBit(PORTB, PB2);
 setBit(DDRB, DDD5);
 clearBit(PORTB, PB1);
}
void initAdc()
{
 ADMUX |= (1 << REFS0); //Ajuste da referencia de tensão
 ADMUX |= (1 << ADSP1) | (1 << ADPS0); //Ajusta o pre scaler para 1/8
 ADMUX |= (1 << ADEN); //Habilita o Hardware do conversor para funcionar
}
void esolhe canal(uint8_t canal)
{
 switch (canal)
 {
   case 0: //Sistema ligado
     setBit(ADMUX, MUX0);
     clearBit(ADMUX, MUX1);
     clearBit(ADMUX, MUX2);
     clearBit(ADMUX, MUX3);
     clearBit(ADMUX, MUX4);
     break;
   case 1: //Umidade baixa
```

```
    setBit(ADMUX, MUX0);
    setBit(ADMUX, MUX1);
    clearBit(ADMUX, MUX2);
    clearBit(ADMUX, MUX3);
    setBit(ADMUX, MUX4);
    break;
   case 2: // Umidade moderada
    setBit(ADMUX, MUX0);
    clearBit(ADMUX, MUX1);
    setBit(ADMUX, MUX2);
    clearBit(ADMUX, MUX3);
    setBit(ADMUX, MUX4);
    break;
   case 3://Umidade alta
    clearBit(ADMUX, MUX0);
    clearBit(ADMUX, MUX1);
    clearBit(ADMUX, MUX2);
    clearBit(ADMUX, MUX3);
    setBit(ADMUX, MUX4);
    break;

 }
}
```

- Sensor_com_biblioteca

```
//Inicialização do conversor ADC - por padrão, vem configurado para o canal 0
void initAdc(void)
{
 ADMUX |= (1 << REFS0);  //Ajuste da tensão de referência
 ADCSRA |= (1 << ADPS1) | (1 << ADPS0); //Ajusta o pre-scaler para 1/8
 ADCSRA |= (1 << ADEN); //Habilita o hardware do conversor para funcionar
}
```

```
void escolheCanal(uint8_t canal)
{
  switch (canal)
  {
    case 0:
      clearBit(ADMUX, MUX0);
      clearBit(ADMUX, MUX1);
      clearBit(ADMUX, MUX2);
      clearBit(ADMUX, MUX3);
      break;
    case 1:
      setBit(ADMUX, MUX0);
      clearBit(ADMUX, MUX1);
      clearBit(ADMUX, MUX2);
      clearBit(ADMUX, MUX3);
      break;
    case 2:
      clearBit(ADMUX, MUX0);
      setBit(ADMUX, MUX1);
      clearBit(ADMUX, MUX2);
      clearBit(ADMUX, MUX3);
      break;
    case 3:
      setBit(ADMUX, MUX0);
      setBit(ADMUX, MUX1);
      clearBit(ADMUX, MUX2);
      clearBit(ADMUX, MUX3);
      break;
    case 4:
      clearBit(ADMUX, MUX0);
      clearBit(ADMUX, MUX1);
```

```
      setBit(ADMUX, MUX2);
      clearBit(ADMUX, MUX3);
      break;
    case 5:
      setBit(ADMUX, MUX0);
      clearBit(ADMUX, MUX1);
      setBit(ADMUX, MUX2);
      clearBit(ADMUX, MUX3);
      break;
    default:
      clearBit(ADMUX, MUX0);
      clearBit(ADMUX, MUX1);
      clearBit(ADMUX, MUX2);
      clearBit(ADMUX, MUX3);
      break;
  }
}
int fazerLeitura(uint8_t canal)
{
  escolheCanal(canal);
  ADCSRA |= (1 << ADSC);  //Liga o ADC no canal atual
  loop_until_bit_is_clear(ADCSRA, ADSC);  //Espera até a conversão terminar
  return  ADC;
}
```

- Usart_biblioteca

```
#include <avr/io.h>
#include <util/delay.h>
#define BAUD 9600
#include <util/setbaud.h>
//Macros utlizados nos programas
#define setBit(byte, bit) (byte |= 1 << bit)
```

```
#define clearBit(byte, bit) (byte &= ~(1 << bit))
#define toggleBit(byte, bit) (byte ^= (1 << bit))
#define isBitSet(byte, bit) ((byte & (1 << bit)) == (1 << bit) ? 1 : 0)
// Configura a USART
void initUSART0(void)
{
 //Define a velocidade de comunicação
 UBRR0H = UBRRH_VALUE;
 UBRR0L = UBRRL_VALUE;
 setBit(UCSR0B, TXEN0); //Habilita a transmissão serial
 setBit(UCSR0B, RXEN0); //Habilita a recepção serial
 setBit(UCSR0C, UCSZ01);//Configura comunicação em 8 bits com 1 stopbit
 setBit(UCSR0C, UCSZ00);
}
//Função para transmitir 1 byte
void transmitByte( uint8_t data)
{
 loop_until_bit_is_set(UCSR0A, UDRE0);
 UDR0 = data;
}
uint8_t receiveByte(void)
{
 loop_until_bit_is_set(UCSR0A, RXC0);
 return UDR0;
}
//Função para transmitir String
void transmitirString(const char myString[])
{
 uint8_t i = 0;
 while (myString[i])
 {
```

```
   transmitByte(myString[i]);
   i++;
  }
}
void setup()
{
  initUSART0();
  setBit(DDRB, DDB5);
  clearBit(PORTB, PORTB5);
}
void loop()
{
  uint8_t x = receiveByte();
  if (x == 'a')
    setBit(PORTB, PORTB5);
  else if (x == 'b')
    clearBit(PORTB, PORTB5);
  else
    transmitirString("Seu besta, conserta essa porra aqui");
}
```

minha_Adc

```
//Inicialização do conversor ADC - por padrão, vem configurado para o canal 0
void initAdc(void)
{
  ADMUX |= (1 << REFS0);  //Ajuste da tensão de referência
  ADCSRA |= (1 << ADPS1) | (1 << ADPS0); //Ajusta o pre-scaler para 1/8
  ADCSRA |= (1 << ADEN); //Habilita o hardware do conversor para funcionar
}
void escolheCanal(uint8_t canal)
{
```

```
switch(canal)
{
  case 0:
    clearBit(ADMUX, MUX0);
    clearBit(ADMUX, MUX1);
    clearBit(ADMUX, MUX2);
    clearBit(ADMUX, MUX3);
    break;
  case 1:
    setBit(ADMUX, MUX0);
    clearBit(ADMUX, MUX1);
    clearBit(ADMUX, MUX2);
    clearBit(ADMUX, MUX3);
    break;
  case 2:
    clearBit(ADMUX, MUX0);
    setBit(ADMUX, MUX1);
    clearBit(ADMUX, MUX2);
    clearBit(ADMUX, MUX3);
    break;
  case 3:
    setBit(ADMUX, MUX0);
    setBit(ADMUX, MUX1);
    clearBit(ADMUX, MUX2);
    clearBit(ADMUX, MUX3);
    break;
  case 4:
    clearBit(ADMUX, MUX0);
    clearBit(ADMUX, MUX1);
    setBit(ADMUX, MUX2);
    clearBit(ADMUX, MUX3);
```

```
      break;
    case 5:
      setBit(ADMUX, MUX0);
      clearBit(ADMUX, MUX1);
      setBit(ADMUX, MUX2);
      clearBit(ADMUX, MUX3);
      break;
    case 6:
      setBit(ADMUX, MUX0);
      clearBit(ADMUX, MUX1);
      clearBit(ADMUX, MUX2);
      setBit(ADMUX, MUX3);
      break;
    case 7:
      setBit(ADMUX, MUX0);
      setBit(ADMUX, MUX1);
      setBit(ADMUX, MUX2);
      clearBit(ADMUX, MUX3);
      break;
    case 8:



    default:
      clearBit(ADMUX, MUX0);
      clearBit(ADMUX, MUX1);
      clearBit(ADMUX, MUX2);
      clearBit(ADMUX, MUX3);
      break;
  }
}
int fazerLeitura(uint8_t canal)
```

```
{
  escolheCanal(canal);
  ADCSRA |= (1 << ADSC);  //Liga o ADC no canal atual
  loop_until_bit_is_clear(ADCSRA, ADSC);  //Espera até a conversão terminar
  return  ADC;
}
```

- minhasCom

```
//definição para o setbaud funcionar
#define BAUD 9600
#include <util/setbaud.h>
//funções de configuração da porta serial
void initUSART(void)
{
   UBRR0H = UBRRH_VALUE;
   UBRR0L = UBRRL_VALUE;
   //Se o multiplicador de clock estiver ligado
   #if USE_2X
      UCSR0A |= (1 << U2X0);
   #else
      UCSR0A &= ~(1 << U2X0);
   #endif

   UCSR0B = (1 << TXEN0) | (1 << RXEN0);   //Habilita a transmissão e recepção de dados
   UCSR0C = (1 << UCSZ01) | (1 << UCSZ00); //Configura a comunicação para 8 bits, 1 stop bit
}
void transmitByte(uint8_t data)
{
   loop_until_bit_is_set(UCSR0A, UDRE0);
   UDR0 = data;
}
uint8_t receiveByte(void){
```

```
   loop_until_bit_is_set(UCSR0A, RXC0);
   return UDR0;
}
void printString(const char myString[])
{
   uint8_t i = 0;
   while(myString[i]){
      transmitByte(myString[i]);
      i++;
   }
}
```

- minhasMacro

```
//Definições de macros utilizadas no projeto
#define setBit(byte, bit) (byte |= 1 << bit)
#define clearBit(byte, bit) (byte &= ~(1<<bit))
#define toggleBit(byte, bit) (byte ^= 1 << bit)
#define isBitSet(byte,bit) ((byte & (1<<bit)) == (1<<bit)? 1 : 0)
```

Referência das imagens

High sensitivity moisture sensor. Visto em: <https://www.fecegypt.com/uploads/dataSheet/1480854383_water%20and%20soil.pdf> Ultima visualização: 24/04/2019

Hack educa. Visto em: <https://www.hackeduca.com.br/fc-28-yl-38-yl-69-sensor-de-umidade-higrometro-sensor-de-solo/> Ultima visualização: 24/04/2019