# Sistema Automatizado para Cultivo de Plantas

## *Controle Umidade do Solo*

### 1. Componentes

Para controlarmos a umidade do solo da planta, iniciamos utilizando o módulo da Figura 01, que se baseia na leitura de umidade de acordo com a diferença de tensão entre as hastes fincadas na terra. Percebemos, depois de utilizarmos o módulo, que estava ocorrendo indícios de corrosão, assim, trocamos para o módulo capacitivo, ilustrado na Figura 02.

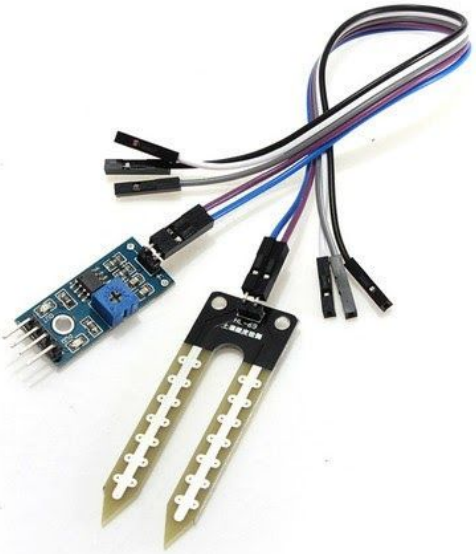


Figura 01 - Sensor de Umidade por Tensão  Figura 02 - Sensor de Umidade Capacitivo

Ambos os módulos fornecem saídas analógicas (a que utilizamos no projeto); a diferença é que primeiro módulo possui também uma saída digital. Tal saída serve para podermos regular um nível de tensão que queremos, através do pequeno potenciômetro que há na placa do módulo, assim, quando tal leitura de tensão fosse atingida, a porta digital daria um sinal (true) que seria lido. Para o projeto, só necessitamos da leitura da tensão, ou seja, a leitura analógica.

### 2. Circuito

O circuito abaixo, Figura 03, mostra como são as ligações entre os módulos e a placa de controle. Note que poderia ser utilizado

qualquer módulo de controle. Para ilustrarmos, utilizamos um Arduino Nano; no projeto real, utilizamos um ESP8266.

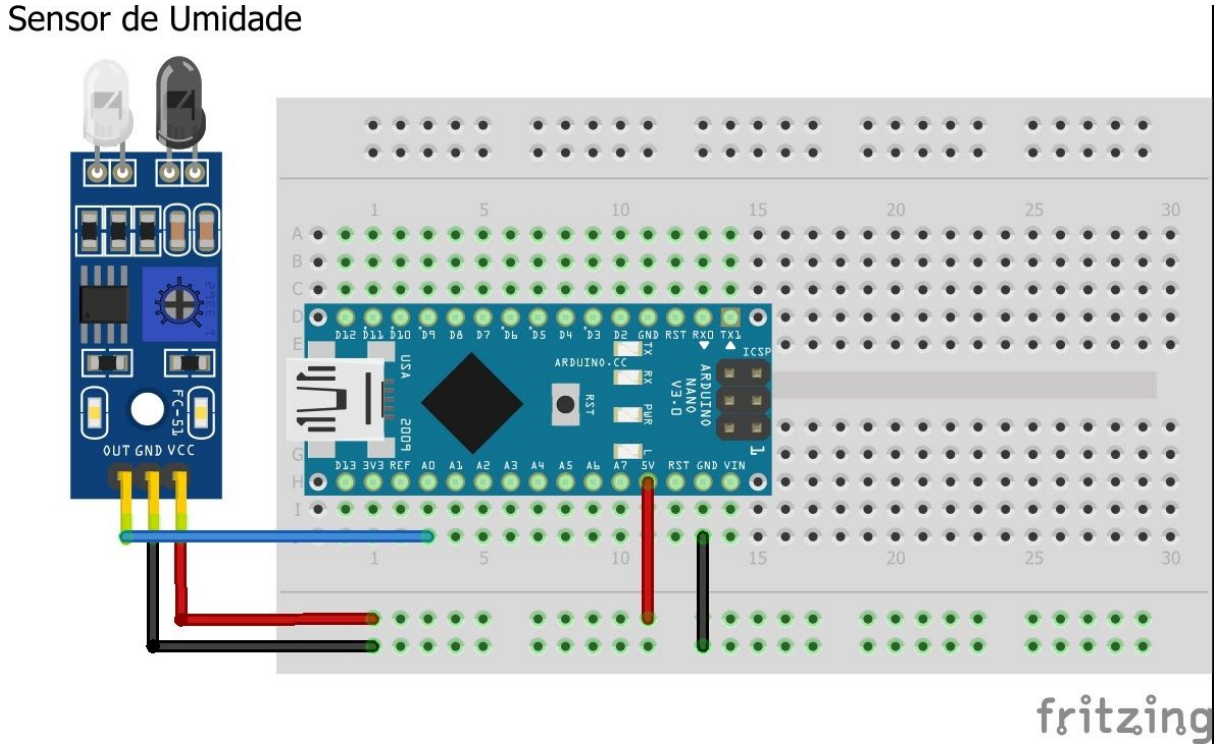


Figura 03 - Circuito Sensor de Umidade

## 3. Código

```
27
28  // Soil Mosture Sensor (SMS) ////////////////////
29  #define SMS A0                          // ESP8266 pin
30  double SMS_Value = 0.0;
31  double SMS_Volts = 0.0;
32  double SMS_Perct = 0.0;
33  double WLS_Cdt = 0.0;                   // Water Level Sensor Condition
34  #define SMS_Dry 3.08                    // Parametros para conversao do valor lido em porcentagem
35  #define SMS_Moist 1.2
36  #define SMS_Perct_Min 40                // Parametro minimo de umidade (controle da valvula)
37  ////////////////////////////////////////////////
38
```

```
86  void loop () {
87
88    // Soil Mosture Sensor Iteration /////////
89    SMS_Value = analogRead(SMS);               // Le o sinal analogico do SMS
90    Serial.print("Soil Mosture Analog Value:"); // Printa valor no monitor serial
91    Serial.println(SMS_Value);
92    // convert the analog signal to voltage
93    // the ESP2866 A0 reads between 0 and ~3 volts, producing a corresponding value
94    // between 0 and 1024. The equation below will convert the value to a voltage value.
95    SMS_Volts = (SMS_Value * 3.08) / 1024;
96    Serial.print("Soil Mosture Analog Voltage:");
97    Serial.println(SMS_Volts);
98
99    int SMS_Perct_Aux = (1 - ((SMS_Volts-SMS_Moist)/(SMS_Dry-SMS_Moist)))*100;
100
101   if (abs(SMS_Perct - SMS_Perct_Aux) >= 5.0) {          // Se a diferenca entre a leitura atual e a anterior for maior que 5
102     SMS_Perct = SMS_Perct_Aux;                          // Atualizamos a leitura
103     sendData = true;                                    // Habilitamos o envio ao banco de dados
104   }
105   ////////////////////////////////////////
106
```

## 4. Lógica

O código que iremos colocar no nosso controlador terá que interpretar o sinal analógico recebido, que é a saída do módulo de leitura de umidade do solo, e realizar alguma ação com esse valor lido.
Na primeira parte do código definimos as constantes que iremos utilizar para manipular essa componente (tudo isso feito antes mesmo de entrar no *setup* ou *loop* do código). A constante *SMS* indica a porta analógia da placa que conectamos o módulo (no caso, porta A0). As variáveis *double* declaradas na sequência e as com *#define* servirão para fazermos as conversões entre os valores lidos de tensão com a umidade do solo (seco ou molhado), trataremos disso mais tarde.
Na segunda parte do código, dentro do *loop*, iniciamos o tratamento do módulo de umidade. Primeiro pegamos o valor lido pelo módulo com a função *analogRead*, que recebe como parâmetro a porta de entrada dos dados analógicos do módulo (porta A0 que definimos como *SMS*),.
Escolhemos imprimir esse valor no monitor serial através da função *Serial.print* e *Serial.println* (a primeira imprime na tela o que queremos, sem pular linha no final; a segunda pula - *ln*).
Seguindo, fazemos a conversão do valor lido na porta digital para um valor entre 0 e *SMS_Dry*, que é a tensão máxima que obtivemos de um solo seco (obtivemos um valor aproximado de 3,08). Assim, em *SMS_Volts* temos a leitura do módulo de umidade em voltagem; na sequência imprimimos esse valor na tela.
Na próxima linha de código fazemos a conversão do valor lido em voltagem para porcentagem (estipulamos valores mínimos e máximos de acordo com testes em solo seco e molhado).
No loop que segue, com *if*, iremos comparar a atual leitura do sensor de umidade *SMS_Perc_Aux* com a ultima leitura feita *SMS_Perc*, se a diferença

da leitura for maior que 5% nós fazemos as linhas seguintes do loop: atualizamos o valor de *SMS_Perc* e habilitamos a variável *sendData* (tal variável só será tratada na parte de interação com Banco de Dados).

## *Controle Válvula Irrigação*

### 1. Componentes

Para conseguirmos controlar a vazão de água para irrigação utilizamos uma válvula solenóide, Figura 04. Seu funcionamento é bem simples: funciona como se fosse um interruptor de luz, se acionarmos ela permite a passagem de água, se desligarmos, ela se fecha. Durante a instalação reparamos que há um sentido certo de entrada da mangueira e saída (há setas indicando o sentido na própria válvula). Além disso, ao ligar a válvula direto na torneira, percebemos que a pressão ajuda a melhorar o funcionamento da válvula (a pressão da rede facilita o abrir e fechar da válvula). Além disso, precisamos de um componente para podermos controlar a válvula, como um botão de liga/desliga, assim, utilizamos um transistor ou relé. A função desse componente é de fornecer a tensão de 12V para a válvula, porém só quando queremos, através de um controle de liga/desliga, que funciona com a mesa tensão de operação da placa do controlador (5V). Ou seja, conseguimos controlar um aparelho que opera a uma tensão de 12V utilizando uma placa que opera em 5V, sem queimar nada, isso ocorre pois o relé/transistor possui um circuito liga/desliga separado daquele que irá alimentar o aparelho. No circuito escolhemos utilizar o transistor TIP120, Figura 05, por ser mais barato e menor que um módulo relé.

Figura 04 - Válvula Solenóide

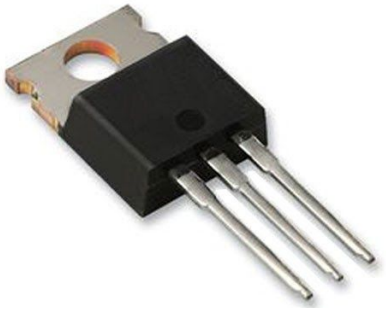

Figura 05 - Transistor TIP120

## 2. Circuito

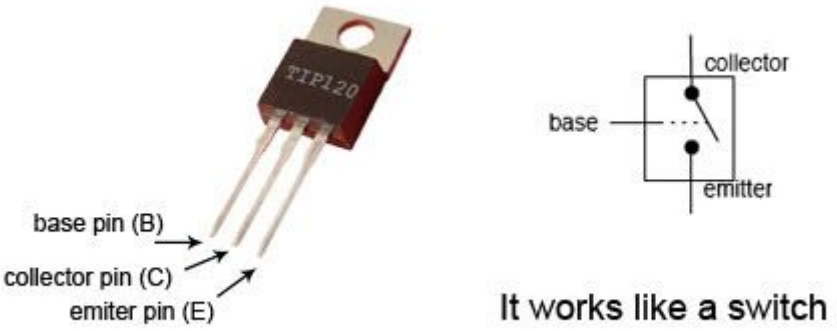


Figura 06 - Transistor TIP120

O primeiro passo é nos atentar para as conexões do transistor: cada perna do transistor possui uma função específica, não entraremos na explicação do circuito interno do transistor, mas basicamente, como ilustrado na Figura 06, da esquerda para a direita, existem os pinos da Base, Coletor e Emissor, basicamente, se mandarmos uma corrente pela base, o coletor curtocircuita com o emissor, ou seja, fecha o circuito, funcionando como um interruptor. Assim, como na Figura 07, ligamos o terra da válvula no coletor e o terra da fonte e da placa no emissor. Assim quando a base acionar o interruptor, o terra da válvula irá ligar, permitindo o seu funcionamento. O diodo entre o terra (GND) e o positivo (VCC) da válvula serve como proteção para o circuito e vale salientar que ele possui um sentido, como o indicado na figura (devido a Tensão Reversa o diodo é colocado em paralelo com a bobina, isso não será explicado nesse projeto). O resistor serve para regular a corrente para o transistor e proteger o circuito; ele é conectado na porta digital da placa que irá controlar o circuito, no caso a D0. A fonte de 12V é conectada em barramentos separados da protoboard (cuidado para não conectar o VCC da fonte na placa, já que controladores geralmente operam em tensões de 5V e a fonte é de 12V).

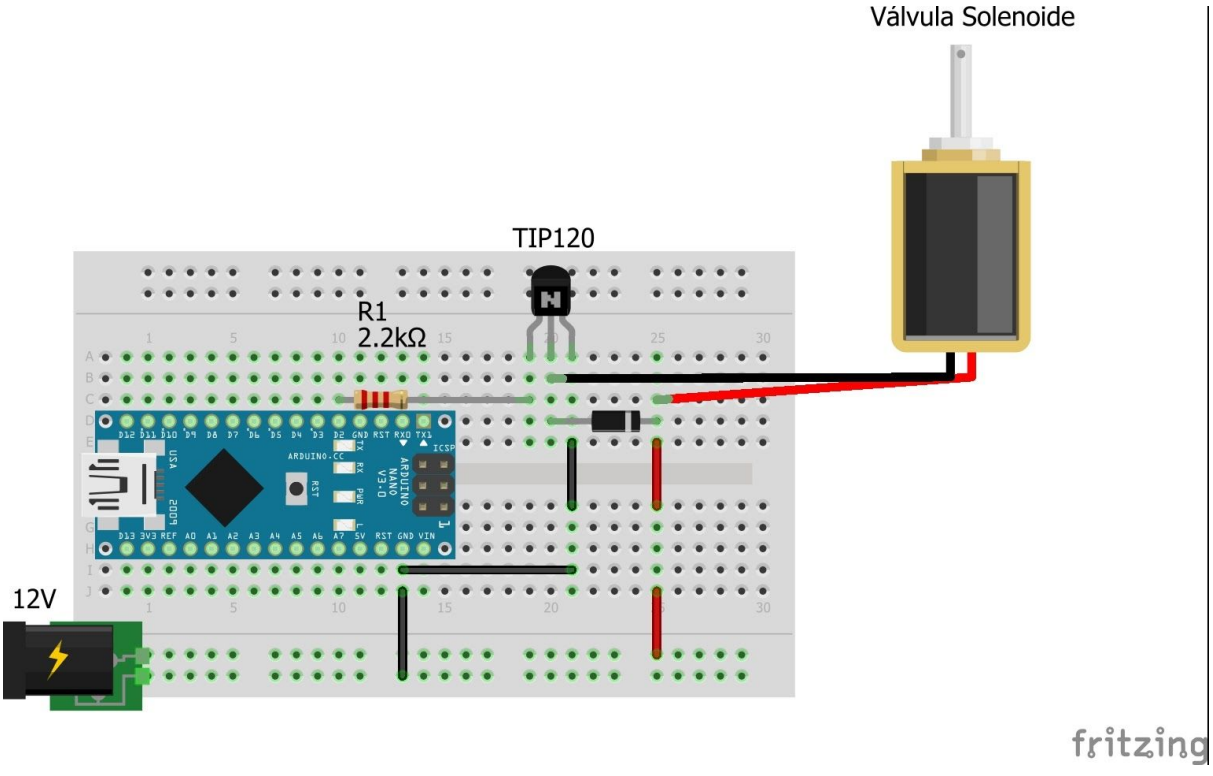

Figura 07 - Circuito Válvula Solenóide

## 3. Código

```
43
44  // Valve ////////////////////////////////////
45  #define VALVE D0                    // Valve pin
46  #define GLED D1                     // Led pin
47  String valveState = "Fechada";      // Estado da valvula
48  ////////////////////////////////////////////
49
```

```
79
80    //Valve ////////////////////////////////
81    pinMode (VALVE, OUTPUT);
82    pinMode (GLED, OUTPUT);
83    ////////////////////////////////////////
```

```
126
127 // Valve Iteration ///////////////////////
128  if (SMS_Perct < SMS_Perct_Min) {  // Se leitura de umidade for inferior ao valor mínimo
129    digitalWrite(VALVE, HIGH);      // Abrimos a valvula
130    digitalWrite(GLED, HIGH);       // Ligamos o led
131    Serial.println("Valve Open");   // Printamos no monitor serial
132    valveState = "Aberta";
133  }
134  else {
135    digitalWrite(VALVE, LOW);
136    Serial.println("Valve Closed");
137    digitalWrite(GLED, LOW);
138    valveState = "Fechada";
139  }
140 ////////////////////////////////////////////
141
```

## 4. Lógica

No início do código, antes do *setup* ou *loop*, declaramos as constantes a serem utilizadas para o tratamento da válvula, no caso criamos as constantes *VALVE* para nos referirmos a porta digital de controle do módulo, a D0, e *GLED* uma saída digital qualquer, D1, para controlarmos um led que indica se a válvula está ligada ou não. Definimos também uma variável do tipo *String* chamada de *valveState* para nos indicar o estado da válvula (inicializamos ela como *“Fechada”*).
Nas próximas linhas de código já estamos dentro do *setup* onde definimos as duas portas declaradas *VALVE* e *GLED* como saídas, através da função *pinMode*.
Depois, já dentro do *loop* inicializamos o tratamento do controle da válvula. Na condição criada, verificamos se a leitura atual da umidade do solo é

inferior a leitura mínima em porcentagem que declaramos (no nosso código estipulamos 40%), assim, se a condição for satisfeita, iniciamos o tratamento: ligamos a válvula e o led com o comando *digitalWrite* e escrevendo *HIGH*; imprimimos na tela o estado da válvula e atualizamos o valor da variável *valveState* para *“Aberta”*. Se a condição não for aceita, *else*, desligamos a válvula e o led “escrevendo” *LOW* na saída analógica, imprimimos na tela e atualizamos a variável *valveState* para *“Fechada”*.