



BEBEDOURO AUTOMÁTICO

OBJETIVO



- ANIMAIS PEQUENOS E MEDIOS
- AUXILIAR O COTIDIANO
- PRESERVAR SAÚDE DOS ANIMAIS
- HIDRATAÇÃO
- CONTROLE
- MONITORAMENTO



COMPONENTES



- 1 MÓDULO ULTRASSÔNICO HC-SR04
- 1 FONTE DE ALIMENTAÇÃO 12V 2ª
- 1 MÓDULO RELÉ 1 CANAL
- 1 MINI BOMBA D'ÁGUA
- 1 ARDUINO UNO
- 1 SENSOR DE NÍVEL DE ÁGUA LATERAL MINI BOIA
- 1 CABO JUMPER MACHO X FÊMEA
- 1 MÓDULO ESP8266
- 1 PROTOBOARD



CÓDIGO



```
#include <SoftwareSerial.h>
```

```
#include <Ultrasonic.h>
```

```
//Definindo Serial de comunicação com Esp8266
```

```
SoftwareSerial ArduinoUno(3,2);
```

```
//Define os pinos conectados com base nas funções
```

```
#define pino_trigger 4
```

```
#define pino_echo 5
```

```
#define bomba 7
```

```
#define agualvl 8
```

```
//Inicializa o sensor de distância nos pinos definidos acima
```

```
Ultrasonic ultrasonic(pino_trigger, pino_echo);
```

```
int ativ = 0;
```

```
void setup(){
```

```
//Inicia A conexão Serial com o Esp8266
```

```
    ArduinoUno.begin(9600);
```

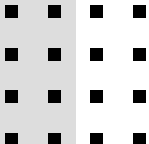
```
    pinMode(2, OUTPUT);
```

```
    pinMode(3, INPUT);
```

```
//Inicia os pinos configurados  
pinMode(bomba, OUTPUT);  
pinMode(aguallvl, INPUT);//captando informação "INPUT"  
}
```

```
//Função para enviar informações para o Esp8266  
void SendInf(String mensagem){  
  //interpreta oque o Esp pediu e envia o Nível de Água  
  if(mensagem.indexOf("aqualvl")>-1){  
    int lvl = digitalRead(aguallvl);  
    ArduinoUno.println("inflvl "+String(lvl));  
  }  
}
```

```
//envia o sinal para o Esp avisando que a bomba foi ativada  
if(mensagem.indexOf("bebeu")>-1){  
  ArduinoUno.println("drink");  
}  
}
```



```
void loop(){
    //Lê as informações do sensor de distância, em cm
    float cmMsec;
    long microsec = ultrasonic.timing();
    cmMsec = ultrasonic.convert(microsec, Ultrasonic::CM);
    //Lê as informações do sensor de nível de água
    int estado = digitalRead(aguaIvl);

    //Limita a ativação da bomba com base na proximidade(sensor de distância) e pelo nível de
    água(sensor de nível de água)
    if(cmMsec < 20 && estado == 1){
        //atrasar a ativação para não ligar sem necessidade e não enviar informação errada
        delay(2000);
        //Re-Lê as informações dos sensores Sensor de distância e de nível de água
        microsec = ultrasonic.timing();
        cmMsec = ultrasonic.convert(microsec, Ultrasonic::CM);
        estado = digitalRead(aguaIvl);
        //verifica se as condições ainda estão batendo
        //se estiverem ativa se não, então não ativa
        if(cmMsec < 20 && estado == 1){
            ativ = 1;
            //Chama a função para informar ao Esp que a bomba foi ativada
            SendInf("bebeu");
        }
    }
}
```

```
//desativando a bomba por fora para ter certeza que ela não ative sozinha derrepente
digitalWrite(bomba, 0);
//ativando a bomba
if(ativ == 1){
    digitalWrite(bomba, 1);
    delay(5000);
    ativ = 0;
    digitalWrite(bomba, 0);
}

//Lendo informações que o Esp manda
if(ArduinoUno.available()>0){
    String val = ArduinoUno.readString();
    //Chamando função para interpretar a "mensagem" recebida
    SendInf(val);
}
}
```




CÓDIGO ESP8266

```
#include <SoftwareSerial.h>
#include <TimeLib.h>
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
```

```
//Definindo Serial de comunicação com Arduino
SoftwareSerial NodeMCU(D2,D3);
```

```
//Informações da rede
const char* ssid    = "Seles";
const char* password = "28029021";
```

```
//Definindo as variaveis e funções de Acesso ao horário em tempo real do Brasil
static const char ntpServerName[] = "br.poolo.ntp.org";
const int timeZone = -3;
WiFiUDP Udp;
unsigned int localPort = 8888;
time_t getNtpTime();
void printDigits(int digits);
void sendNTPpacket(IPAddress &address);
```

```
//Porta do WebServer no roteador
WiFiServer server(80);
```

// Variavel criando o cabeçalho

String header;

// Variaveis para organizar a comunicação Arduino-Esp e Esp-Site

int lvl = 1;

int nvl = 1;

int agua;

String bebeu = "";

//Variaveis de "tempo de comunicação" do esp com o site

unsigned long currentTime = millis();

unsigned long previousTime = 0;

const long timeoutTime = 2000;

void setup() {

 Serial.begin(115200);

//Iniciando o Serial de comunicação com o Arduino e definindo as portas de comunicação

 NodeMCU.begin(9600);

 pinMode(D2,INPUT);

 pinMode(D3,OUTPUT);

//Conectar ao WiFi com Nome e Senha da Rede

 Serial.print("Connecting to ");

 Serial.println(ssid);

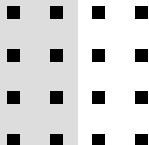
 WiFi.begin(ssid, password);

 while (WiFi.status() != WL_CONNECTED) {

 delay(500);

 Serial.print(".");

 }



```
Serial.println("");
Serial.println("WiFi connected.");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
server.begin();
```

//Iniciando Horário em tempo real

```
Udp.begin(localPort);
setSyncProvider(getNtpTime);
setSyncInterval(300);
}
```

//Função para tratar das informações recebidas pelo esp através do Serial do Arduino

```
void GetInf(String oque){
    //Nível de água Atual
    if(oque.indexOf("inflvl")>-1){
        agua = oque.substring(7).toInt();
    }
    //Última vez em que a bomba foi ativada
    if(oque.indexOf("drink")>-1){
        bebeu = "";
        if (hour() < 10){
            bebeu += "0";
            bebeu += hour();
        }else bebeu += hour();
        bebeu += ":";
        if (minute() < 10){
            bebeu += "0";
            bebeu += minute();
        }else bebeu += minute();
    }
}
```

```
}else bebeu+=minute();
    bebeu+=":";
    if (second() < 10){
        bebeu+="0";
        bebeu += second();
    }else bebeu+=second();

    bebeu+=(" ");
    bebeu+=day();
    bebeu+=".";
    bebeu+=month();
    bebeu+=".";
    bebeu+=year();

}
}
```

```

void loop(){
    //Criando e atualizando página
    WiFiClient client = server.available();
    if (client) {
        String currentLine = "";
        currentTime = millis();
        previousTime = currentTime;
        while (client.connected() && currentTime - previousTime <= timeoutTime) {
            currentTime = millis();
            if (client.available()) {
                char c = client.read();
                header += c;
                if (c == '\n') {
                    if (currentLine.length() == 0) {
                        client.println("HTTP/1.1 200 OK");
                        client.println("Content-type:text/html");
                        client.println("Connection: close");
                        client.println();
                        //Executando comando quando a pagina /waterlvl for adicionada no cabeçalho
                        //da página ou quando o botão "Checar nivel" for pressionado
                        if (header.indexOf("GET /waterlvl") >= 0) {
                            NodeMCU.println("aqualvl");//Mandar mensagem para o arduino dizendo que quer o nivel de
                            agua atual
                            delay(50);
                            Getlnf(NodeMCU.readString());//Recebe a mensagem do arduino dizendo qual o nivel de agua
                            atual
                            delay(100);
                            lvl = 2; //Controle de ativação da mudança da página para mostrar o nivel de água
                        }else if (header.indexOf("GET /timeon") >= 0) {
                            nvl = 2; //Controle de ativação da mudança da página para mostrar a última vez que a bomba
                            foi ativada
                        }
                    }
                }
            }
        }
    }
}

```

//Mostrar a página HTML

```
client.println("<!DOCTYPE html><html>");
client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
client.println("<link rel=\"icon\" href=\"data:;\">");
```

//Criando o design do Botão

```
client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}");
client.println(".button { background-color: #195B6A; border: none; color: white; padding: 16px 400px;");
client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");
client.println(".button2 {background-color: #77878A;}</style></head>");
```

//"Titulo" Da Página

```
client.println("<body><h1>BEBEDOURO AUTOMATICO</h1>");
```

//Mensagem

```
client.println("<p>Checar nivel da agua</p>");
```

//CRIAR BOTÃO QUE CHECA O NIVEL DE ÁGUA

```
client.println("<p><a href=\"/waterlvl\"><button class=\"button\">Checar  
Nivel</button></a></p>");
```

//Ativação quando o botão é pressionado

```
if(lvl == 2){
```

```
  //Mostra qual é o nivel da água que foi recebido pelo arduino
```

```
  if(agua == 1)
```

```
    client.println("O nível de água está alto!");
```

```
  else if(agua == 0)
```

```
    client.println("O nível de água está baixo!");
```

```
  lvl = 1;
```

```
}
```

```

//mensagem
    client.println("<p>Checar ultima vez em que foi ativado</p>");
    //CRIAR BOTÃO QUE DIZ A ULTIMA VEZ QUE A BOMBA FOI ATIVADA
    client.println("<p><a href=\"/timeon\"><button class=\"button\">Checar
Ativacao</button></a></p>");
    //Ativação quando o botão é pressionado
    if(nvl == 2){
        //Mostra a informação guardada
        client.println("Ativado a ultima vez as: "+bebeu);
        //reseta a variavel para que possa ser mostrado novamente
        nvl = 1;
    }
    //final da página
    client.println("</body></html>");
    client.println();
    break;
} else {
    currentLine = "";
}
} else if (c != '\r') {
    currentLine += c;
}
}
}

```



```
header = "";
  client.stop();
}
//Lendo as informações que o Arduino Manda
if(NodeMCU.available()>0){
  String val = NodeMCU.readString();
  //Chama a função de tratamento das informações
  GetInf(val);
}
}
```

```
//Código que conecta e armazena o horario em tempo real "NTP" nome do método
/*----- NTP code -----*/
```

```
const int NTP_PACKET_SIZE = 48; // NTP time is in the first 48 bytes of message
byte packetBuffer[NTP_PACKET_SIZE]; //buffer to hold incoming & outgoing packets
```

```
time_t getNtpTime()
{
  IPAddress ntpServerIP;
  while (Udp.parsePacket() > 0) ;
  WiFi.hostByName(ntpServerName, ntpServerIP);
  sendNTPpacket(ntpServerIP);
  uint32_t beginWait = millis();
  while (millis() - beginWait < 1500) {
    int size = Udp.parsePacket();
    if (size >= NTP_PACKET_SIZE) {
      Udp.read(packetBuffer, NTP_PACKET_SIZE);
```

```

unsigned long secsSince1900;
    secsSince1900 = (unsigned long)packetBuffer[40] << 24;
    secsSince1900 |= (unsigned long)packetBuffer[41] << 16;
    secsSince1900 |= (unsigned long)packetBuffer[42] << 8;
    secsSince1900 |= (unsigned long)packetBuffer[43];
    return secsSince1900 - 2208988800UL + timeZone * SECS_PER_HOUR;
}
}
return 0;
}

```

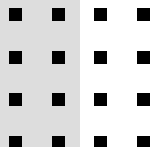
```

void sendNTPpacket(IPAddress &address)
{
    memset(packetBuffer, 0, NTP_PACKET_SIZE);
    packetBuffer[0] = 0b11100011;
    packetBuffer[1] = 0;
    packetBuffer[2] = 6;
    packetBuffer[3] = 0xEC;
    packetBuffer[12] = 49;
    packetBuffer[13] = 0x4E;
    packetBuffer[14] = 49;
    packetBuffer[15] = 52;
    Udp.beginPacket(address, 123);
    Udp.write(packetBuffer, NTP_PACKET_SIZE);
    Udp.endPacket();
}

```

CONCLUSÃO

- O projeto propõe automatizar a forma como é disponibilizada a água para os animais domésticos em residências em que os tutores não permanecem por muito tempo e assim facilitar e melhorar a forma de vida do animal.





FIM

