RELATÓRIO LABORATORIAL

	Dados de Identificação	
Comp. curricular:	ECT2521-PROJETOS COM MICROCONTROLADORES E MINICOMPUTADORES	
	2015012516 20170037950	Leonardo Moniz Sodré Lopes Teixeira Vinícius dos Santos Tavares

Introdução

Neste projeto temos o objetivo de implementar um instrumento musical utilizando o sensor de luminosidade LDR. A partir desse sensor iremos capturar a variação da luminosidade gerada ao passar a mão por cima do sensor, esse dado será interpretado pelo sistema de controle alocado no ESP, que ao receber essa variação emitira o som de uma nota através de uma caixa de som, colocamos um potenciômetro para definir sua frequência, consequentemente o músico define a oitava da nota para tocar. Como para esse experimento iremos usar apenas dois LDR, teremos duas notas e suas oitavas. Para fins de iteração com o público utilizamos o protocolo MQTT para que o som que o músico estiver tocando seja escutado em tempo real por outro ESP que possui a caixa de som, e por sua vez estará alocado em qualquer lugar com internet.

Materiais

Utilizamos neste projeto dois LDR, um potenciômetro, um ESP32, um ESP8266, e dois autofalantes.

Desenvolvimento

Na primeira tentativa de formulação do projeto, tentamos fazê-lo utilizando um único Arduino que funcionaria tanto como interpretador dos dados como também no processo de emissão do som.

Nos primeiros testes estávamos utilizando o código abaixo:

```
int L1 = 10;
int L2 = 8;

int lr1 = A2;
int lr2 = A1;

int pot = A4;

int a[100][4];
int aux = 0;

int but = 5;

int caixa1 = 6;
int caixa2 = 9;
void setup () {
  pinMode (L1, OUTPUT);
  pinMode (L2, OUTPUT);
  pinMode (lr1, INPUT);
  pinMode (lr2, INPUT);
  pinMode (pot, INPUT);
  pinMode (caixa1, OUTPUT);
  pinMode (caixa2, OUTPUT);
  Serial.begin(9600);
}

int luz1, luz2, p, estado = 0;
unsigned int ton;
int estadoluz1 = 0, estadoluz2 = 0;
unsigned int tNota1, tNota2;
unsigned int tPausa =0;
int estadoP = 0;
void loop () {
  p = analogRead (pot);
  p = map (p, 0, 1010, 0, 8);


  luz1 = analogRead (lr1);
```

```
if (luz1 > 960) {
    ton = 44;

    tone (caixa1, ton * pow (2, p));
    estadoluz1 = 1;
    if (butL ()) {
      tNota1 = millis ();
      if(estadoP==0) {
        estadoP=1;
        tPausa = millis () - tPausa;
        }
    }

  } else if (estadoluz1 == 1) {

    noTone(caixa1);
    estadoluz1 = 0;
    if (butL ()) {
      tNota1 = millis () - tNota1;
      gravar (caixa1,44* pow (2, p),
tNota1, tPausa);
      if (estadoP ==1) {
        estadoP=0;
        tPausa = millis ();
        }
    }
  }
```

```
luz2 = analogRead(lr2);

  if (luz2 > 900) {
    ton = 62;

    tone (caixa2, ton * pow (2, p));


    estadoluz2 = 1;
    if (butL ()) {
      tNota2 = millis ();
      if(estadoP==0) {
        estadoP=1;
        tPausa = millis () - tPausa;
        }
    }
  } else if (estadoluz2 == 1) {

    noTone(caixa2);
    estadoluz2 = 0;
    if (butL ()) {
      tNota2 = millis () - tNota1;
      gravar (caixa2,62* pow (2, p),
tNota2, tPausa);
      if (estadoP ==1) {
        estadoP=0;
        tPausa = millis ();
        }
    }
  }
}
```

```
bool butL() {
  if (estado == 0 && !digitalRead(but)) {
    delay (200);
    estado = 1;
  }
  if (estado == 1 && digitalRead(but)) {
    estado = 2;
    return true;
  }
  if (estado == 2 && !digitalRead(but)) {
    delay(200);
    estado = 3;
  }
  if (estado == 3 && digitalRead(but)) {
    estado = 0;
    return false;
  }
}

void gravar (int porta, unsigned int fre, unsigned int timerNota, unsigned int timerP) {
  a[aux][0] = porta;
  a[aux][1] = fre;
  a[aux][2] = timerNota;
  a[aux][3] = timerP;
  aux++;
}
```

Com a utilização de apenas um Arduino para executar toda as funções, tivemos diversos problemas de interferência no circuito. Para solucionar esse problema tentamos isolar a parte de controle em relação aos autofalantes (utilizando duas protoboards), mas sem êxito. O ruído do circuito fazia com que os autofalantes emitissem sons em frequências não desejadas, e até mesmo emitir som quando não eram acionados.

Com esse problema em mente e a ideia de tocar um instrumento de qualquer lugar, tivemos a ideia de utilizar os ESPs para poder fazer uma comunicação a distância e utilizar dos sistemas de controle isolados, um só para os *inputs* e outro só para os *outputs*. Para estabelecer essa comunicação, decidimos utilizar o protocolo MQTT que é um protocolo amplamente utilizado em IOT (*Internet of Things*), baseado no formato *publisher and subscriber*, em que um dispositivo publica as informações e outro subscreve para receber as informações publicadas em um tópico.

Para essa nova implementação desenvolvemos dois novos códigos, um de *publisher*:

```
#include <WiFi.h>
#include <PubSubClient.h>

int lr1 = 34;
int lr2 = 32;
int pot = 33;
int a[100][4];
int aux = 0;

const char* SSID = "Leonardo";
const char* PASSWORD =
"12345678";

WiFiClient wifiClient;


const char* BROKER_MQTT =
"test.mosquitto.org
int BROKER_PORT = 1883;

#define ID_MQTT  "ppsadiohnk"

#define TOPIC_PUBLISH "pp"
PubSubClient MQTT(wifiClient);
   Instancia o Cliente MQTT
passando o objeto espClient

//Declaração das Funções
void mantemConexoes
//Garante que as conexoes com
WiFi e MQTT Broker se
mantenham ativas
void conectaWiFi();     //Faz
conexão com WiFi
void conectaMQTT();     //Faz
conexão com Broker MQTT
void enviaPacote();
```

```
void setup() {

  pinMode(lr1, INPUT);
  pinMode(lr2, INPUT);
  pinMode(pot, INPUT);
  conectaWiFi();

MQTT.setServer(BROKER_MQT
T, BROKER_PORT);
  Serial.begin(9600);


}
int luz1, luz2, p;

void loop() {

mantemConexoes();

enviaValores();
 MQTT.loop();

}
void mantemConexoes() {
  if (!MQTT.connected()) {
    conectaMQTT();
  }

  conectaWiFi(); //se não há
conexão com o WiFI, a conexão é
refeita
}
```

```
void conectaWiFi() {

  if (WiFi.status() ==
WL_CONNECTED) {
    return;
  }

  Serial.print("Conectando-se na
rede: ");
  Serial.print(SSID);
  Serial.println("  Aguarde!");

  WiFi.begin(SSID, PASSWORD);
// Conecta na rede WI-FI
  //WiFi.begin(SSID); // Conecta na
rede WI-FI

  while (WiFi.status() !=
WL_CONNECTED) {
    delay(100);
    Serial.print(".");
  }

  Serial.println();
  Serial.print("Conectado com
sucesso, na rede: ");
  Serial.print(SSID);
  Serial.print("  IP obtido: ");
  Serial.println(WiFi.localIP());
}

void conectaMQTT() {
  while (!MQTT.connected()) {
    Serial.print("Conectando ao
Broker MQTT: ");

Serial.println(BROKER_MQTT);
    if (MQTT.connect(ID_MQTT))
{
      Serial.println("Conectado ao
Broker com sucesso!");
    }
    else {
      Serial.println("Noo foi possivel
se conectar ao broker.");
      Serial.println("Nova tentatica
de conexao em 10s");
      delay(10000);
    }
  }
```

```
String conversor() {
  String  v1, v2, v3;
  p = analogRead(pot);
  p = map(p, 100, 4070, 1, 8);
  luz1 = analogRead(lr1);
  luz2 = analogRead(lr2);
  Serial.print(luz1);
  Serial.print(" ");
  Serial.print(luz2);
  Serial.print(" ");
  if (luz1 > 200)
  {
    v1 = "on";
    v2 = "off";
  }
  if (luz2 > 200) {
    v1 = "off";
    v2 = "on";
  }

  if (luz2 > 200 && luz1 > 200) {
    v1 = "on";
    v2 = "on";
  }
  if (luz2 < 200 && luz1 < 200) {
    v1 = "off";
    v2 = "off";
  }

  v3 = String(p);
  String json[] = {"ldr1:", v1,
",ldr2:", v2, ",f:", v3, "."};

  String msg;
  for (int i = 0; i < 7; i++) {
    msg += json[i];
  }
  return msg;
}
```

```
void enviaValores() {
String msg=conversor();
  const char* dados = msg.c_str();
  Serial.println(msg);
  delay(300);
MQTT.publish(TOPIC_PUBLISH,
dados);
delay(300);

}
```

E outro de *subscriber*:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

//WiFi
const char* SSID = "Leonardo";
// SSID / nome da rede WiFi que deseja se conectar
const char* PASSWORD = "12345678";  z
 // Senha da rede WiFi que deseja se conectar
WiFiClient wifiClient;

//MQTT Server
const char* BROKER_MQTT = "test.mosquitto.org";
//URL do broker MQTT que se deseja utilizar
int BROKER_PORT = 1883;
// Porta do Broker MQTT

//Som
int s1 = D5;
int s2 = D6;

#define ID_MQTT "padiohnk4654121564"
//Informe um ID unico e seu. Caso sejam usados IDs repetidos a ultima conexão irá sobrepor a anterior.
#define TOPIC_PUBLISH "pp"
//Informe um Tópico único. Caso sejam usados tópicos em duplicidade, o último irá eliminar o anterior.
PubSubClient MQTT(wifiClient);
// Instancia o Cliente MQTT passando o objeto espClient

//Declaração das Funções
void mantemConexoes
//Garante que as conexoes com WiFi e MQTT Broker se mantenham ativas
void conectaWiFi();

void conectaMQTT();

void enviaPacote();
```

```
void setup() {
  Serial.begin(9600);
  pinMode(s1,OUTPUT);
  pinMode(s2,OUTPUT);

  conectaWiFi();

MQTT.setServer(BROKER_MQTT, BROKER_PORT);
  MQTT.setCallback(callback);
  mantemConexoes();
  mantemConexoes();
  MQTT.subscribe("pp");
}

void loop() {
  MQTT.loop();
}
void mantemConexoes() {
  if (!MQTT.connected()) {
    conectaMQTT();
  }

  conectaWiFi(); //se não há conexão com o WiFI, a conexão é refeita
}
```

```
void conectaWiFi() {

  if (WiFi.status() == WL_CONNECTED) {
    return;
  }

  Serial.print("Conectando-se na rede: ");
  Serial.print(SSID);
  Serial.println("  Aguarde!");

  WiFi.begin(SSID, PASSWORD); // Conecta na rede WI-FI
  //WiFi.begin(SSID); // Conecta na rede WI-FI

  while (WiFi.status() != WL_CONNECTED) {
    delay(100);
    Serial.print(".");
  }

  Serial.println();
  Serial.print("Conectado com sucesso, na rede: ");
  Serial.print(SSID);
  Serial.print("  IP obtido: ");
  Serial.println(WiFi.localIP());
}
void conectaMQTT() {
  while (!MQTT.connected()) {
    Serial.print("Conectando ao Broker MQTT: ");

Serial.println(BROKER_MQTT);
    if (MQTT.connect(ID_MQTT))
{
      Serial.println("Conectado ao Broker com sucesso!");

    }
    else {
      Serial.println("Não foi possivel se conectar ao broker.");
      Serial.println("Nova tentatica de conexao em 10s");
      delay(10000);
    }
  }
}
```

```
void callback(char* topic, byte*message, unsigned int length) {

  Serial.print("Message arrived on topic: ");
  Serial.print(topic);
  Serial.print(". Message: ");
  String messageTemp;

  for (int i = 0; i < length; i++) {
    Serial.print((char)message[i]);
    messageTemp += (char)message[i];
  }
  Serial.println();
  Serial.println(messageTemp);
  Serial.println();
  String a = messageTemp.substring(messageTemp.indexOf(":")+1);
  String ldr1 = a.substring(0,a.indexOf(","));
  Serial.println(a);
  Serial.print("LDR1: ");
  Serial.println(ldr1);
  a = a.substring(a.indexOf(":")+1);
  String ldr2 = a.substring(0,a.indexOf(","));
  Serial.print("LDR2: ");
  Serial.println(ldr2);
  a = a.substring(a.indexOf(":")+1);
  String mult = a.substring(0,a.indexOf("."));
  Serial.print("Mult: ");
  Serial.println(mult);
  int f1 = 200, f2=500;
  f1 = f1*mult.toInt();
  f2 = f2*mult.toInt();
  exc(ldr1,ldr2,f1,f2);
}

void exc(String l1,String l2,int f1,int f2){
    if(l1=="on"){
      Serial.println("on1");
      tone(s1,f1,1000);
      }
     if(l2=="on"){
      Serial.println("on2");
      tone(s2,f2,1000);
      }

}
```

O *publisher* recebe dados de dois ldrs que são responsáveis de informar qual frequência desejada que saia (cada autofalante possui sua frequência), além de receber dados de um potenciômetro que identifica um multiplicador para essa frequência, gerando assim diferentes escalas, interpreta esses dados e envia uma String por mqtt, que codifica o acionamento de qual caixa e o multiplicador.

Já o *subscriber* recebe os dados do mqtt e interpreta, gerando o acionamento de um dos autofalantes e determinando a escala de frequência que vai emitir o som a partir do multiplicador enviado pelo *publisher*.

Conclusão

Com este projeto aprendemos muitas coisas relacionadas à Arduino, ESP, protocolos de comunicação para aplicação em IOT. No Arduino aprendemos a controlar sons, receber informações de sensores, entre outras coisas, no ESP aprendemos que ele é semelhante ao Arduino em alguns pontos, como por exemplo: ele pode ser programado na mesma IDE, o controle dos sensores é muito parecido, em outros pontos ele é melhor, na comunicação com a internet, pois ele tem WIFI embutido, facilitando o hardware e software, a capacidade de processamento de é maior e o range da entrada analógica com 12 bits em relação ao Arduino que é 10bits, no protocolo de comunicação utilizamos o mqtt pois ele tem uma resposta mais rápida, em relação aos outros que conhecíamos. Com isso e após alguns testes e os problemas apresentados resolvidos, obtemos êxito na validação desse projeto.

Referências

Language Reference. Disponível em:<https://www.arduino.cc/reference/en/>. Acesso em: 21 de nov. de 2019.

Controle e Monitoramento IoT com NodeMCU e MQTT. Disponível em:

<https://www.filipeflop.com/blog/controle-monitoramento-iot-nodemcu-e-mqtt/>. Acesso em: 21 de nov. de 2019.