UNIVERSIDADE FEDERAL DO ESTADO DO PIAUI

UFPI CAMPUS MINISTRO PETRÔNIO PORTELA

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO - CCN

JOÃO RODRIGUES DE MELO NETO

THIAGO SANTOS BRITO

Universidade Federal do Piauí – UFPI

TRABALHO FINAL ARDUINO TRADUTOR DE CÓDIGO MORSE

Teresina - PI

5 de dezembro de 2019

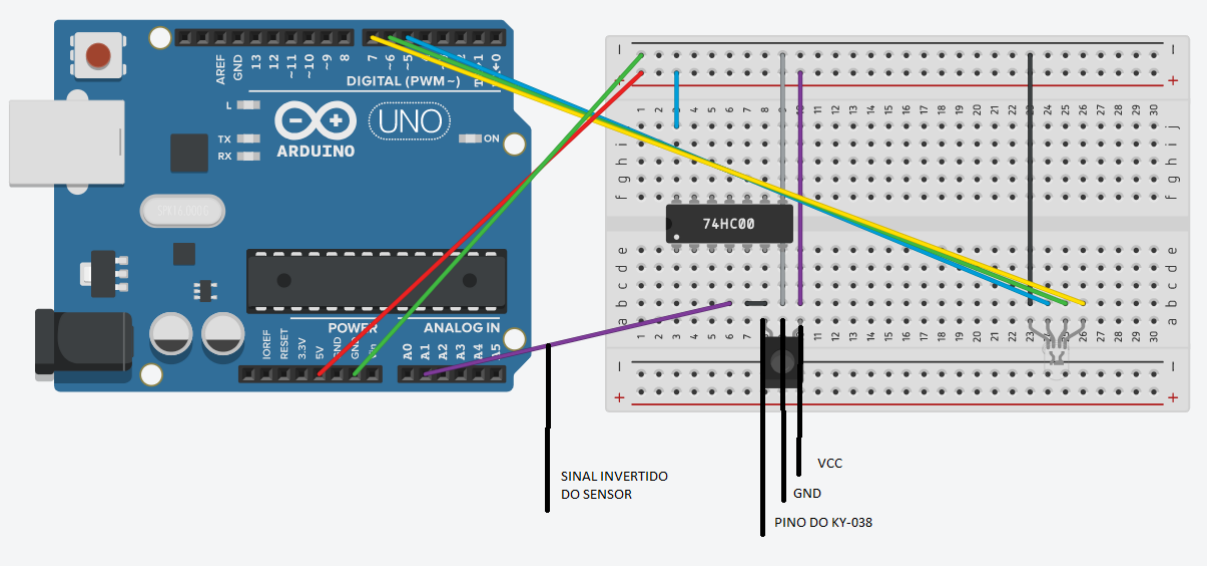
OBJETIVOS

O projeto visa a realização de um leitor e tradutor de código Morse utilizando um Arduino a partir de um código baseado no algoritmo de Goertzel, com a funcionalidade específica de comandar, a partir da decodificação da mensagem criptografada, uma luz LED RGB, com margens no código para implementar uma densa variedade de outros atuadores.

COMPONENTES

- Um Arduino UNO;
- 13 Jumpers M-M;
- Um LED RGB com placa integrada (dispensa resistores);
- Um circuito integrado CD4011 (Porta NAND) – (caso possua um sensor de som com saída analógica, dispensa o uso);
- Protoboard 400 pontos;
- Módulo sensor de som KY-038.

PINAGEM/CIRCUITO



DIFICULDADES

O primeiro empecilho do projeto foi encontrar uma forma de contornar a própria limitação da natureza de funcionamento do sensor de som KY-038, que funciona da seguinte forma: o sensor permanece sempre em alta tensão (HIGH) e quando detecta vibrações, abre o circuito ficando em estado de baixa tensão

(LOW). Para que fosse funcional como um leitor de código Morse e efetivamente captar as interrupções e continuidades da linguagem precisamente como um microfone, foi necessário a utilização do circuito integrado CD4011, uma porta NAND, para fazê-lo atuar de forma contrária, isto é, quando detectasse o som entraria em estado de alta tensão (HIGH). Ainda assim, era necessário calibrar a recepção do sensor. Para isso foi utilizado o algoritmo de Goertzel para transformar a frequência continua e real gerada pelo sensor em dados computacionais e realizar as funções de decodificar o código Morse, isto é, era necessário que o sensor fosse capaz de captar as continuidades para a tradução adequada do código. Por fim, se fez necessária um código de calibração para o sensor funcionar adequadamente distinguindo os sons. Depois de resolvido o problema da captação dos sons, bastaram várias funções que fazem a tradução do Morse para a tabela ASCII e escrita.

CÓDIGO

```
int pinoSensor = A1;

int pinoVermelho = 5;
int pinoVerde = 6;
int pinoAzul = 7;
int soma = 0;
int ledAtual = 0;


float magnitude;
int magnitudelimit = 100;
int magnitudelimit_low = 100;
int realstate = LOW;
int realstatebefore = LOW;
int filteredstate = LOW;
int filteredstatebefore = LOW;


float coeff;
float Q1 = 0;
float Q2 = 0;
float sine;
float cosine;
float sampling_freq=8928.0;
float target_freq=558.0;
float n=48.0;
int testData[48];

int nbtime = 6;

long starttimehigh;
long highduration;
long lasthighduration;
```

```
long hightimesavg;
long lowtimesavg;
long startttimelow;
long lowduration;
long laststarttime = 0;

char code[20];
int stop = LOW;
int wpm;



void setup() {

	// Calculo de goertzel
	int  k;
	float omega;
	k = (int) (0.5 + ((n * target_freq) / sampling_freq));
	omega = (2.0 * PI * k) / n;
	sine = sin(omega);
	cosine = cos(omega);
	coeff = 2.0 * cosine;


	Serial.begin(9600);
	pinMode(pinoVermelho,OUTPUT);
	pinMode(pinoVerde,OUTPUT);
	pinMode(pinoAzul,OUTPUT);

}

void loop() {


	//TOM
	for (char index = 0; index < n; index++)
	{
		testData[index] = analogRead(pinoSensor);
	}
	for (char index = 0; index < n; index++){
		float Q0;
		Q0 = coeff * Q1 - Q2 + (float) testData[index];
		Q2 = Q1;
		Q1 = Q0;
	}
	float magnitudeSquared = (Q1*Q1)+(Q2*Q2)-Q1*Q2*coeff;
	magnitude = sqrt(magnitudeSquared);
	Q2 = 0;
	Q1 = 0;

	if (magnitude > magnitudelimit_low){
		magnitudelimit = (magnitudelimit +((magnitude -
magnitudelimit)/6));
	}

	if (magnitudelimit < magnitudelimit_low)
		magnitudelimit = magnitudelimit_low;
```

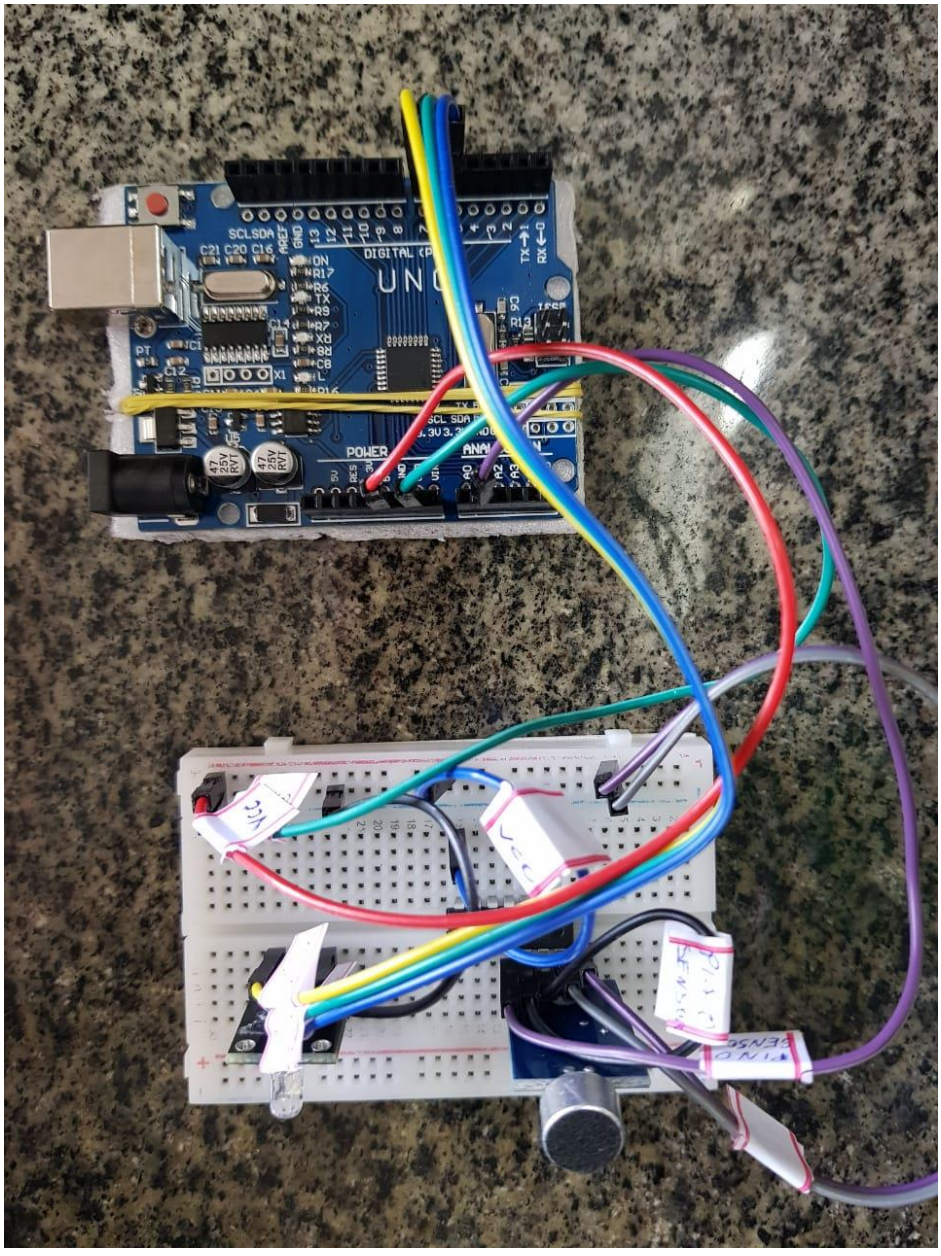
```
            if(magnitude > magnitudelimit*0.6)
                  realstate = HIGH;
            else
                  realstate = LOW;


            if (realstate != realstatebefore){
                  laststarttime = millis();
            }
            if ((millis()-laststarttime)> nbtime){
                  if (realstate != filteredstate){
                        filteredstate = realstate;
                  }
            }

            if (filteredstate != filteredstatebefore){
                  if (filteredstate == HIGH){
                        starttimehigh = millis();
                        lowduration = (millis() - startttimelow);
                  }

                  if (filteredstate == LOW){
                        startttimelow = millis();
                        highduration = (millis() - starttimehigh);
                        if (highduration < (2*hightimesavg) || hightimesavg ==
0){
                              hightimesavg =
(highduration+hightimesavg+hightimesavg)/3;
                        }
                        if (highduration > (5*hightimesavg) ){
                              hightimesavg = highduration+hightimesavg;
                        }
                  }
            }


            if (filteredstate != filteredstatebefore){
                  stop = LOW;
                  if (filteredstate == LOW){
                        if (highduration < (hightimesavg*2) && highduration >
(hightimesavg*0.6)){
                              strcat(code,".");
                        }
                        if (highduration > (hightimesavg*2) && highduration <
(hightimesavg*6)){
                              strcat(code,"-");
                              wpm = (wpm + (1200/((highduration)/3)))/2;
                        }
                  }

                  if (filteredstate == HIGH){

                        float lacktime = 1;
                        if(wpm > 25)lacktime=1.0;
                        if(wpm > 30)lacktime=1.2;
                        if(wpm > 35)lacktime=1.5;

                        //espaco
```

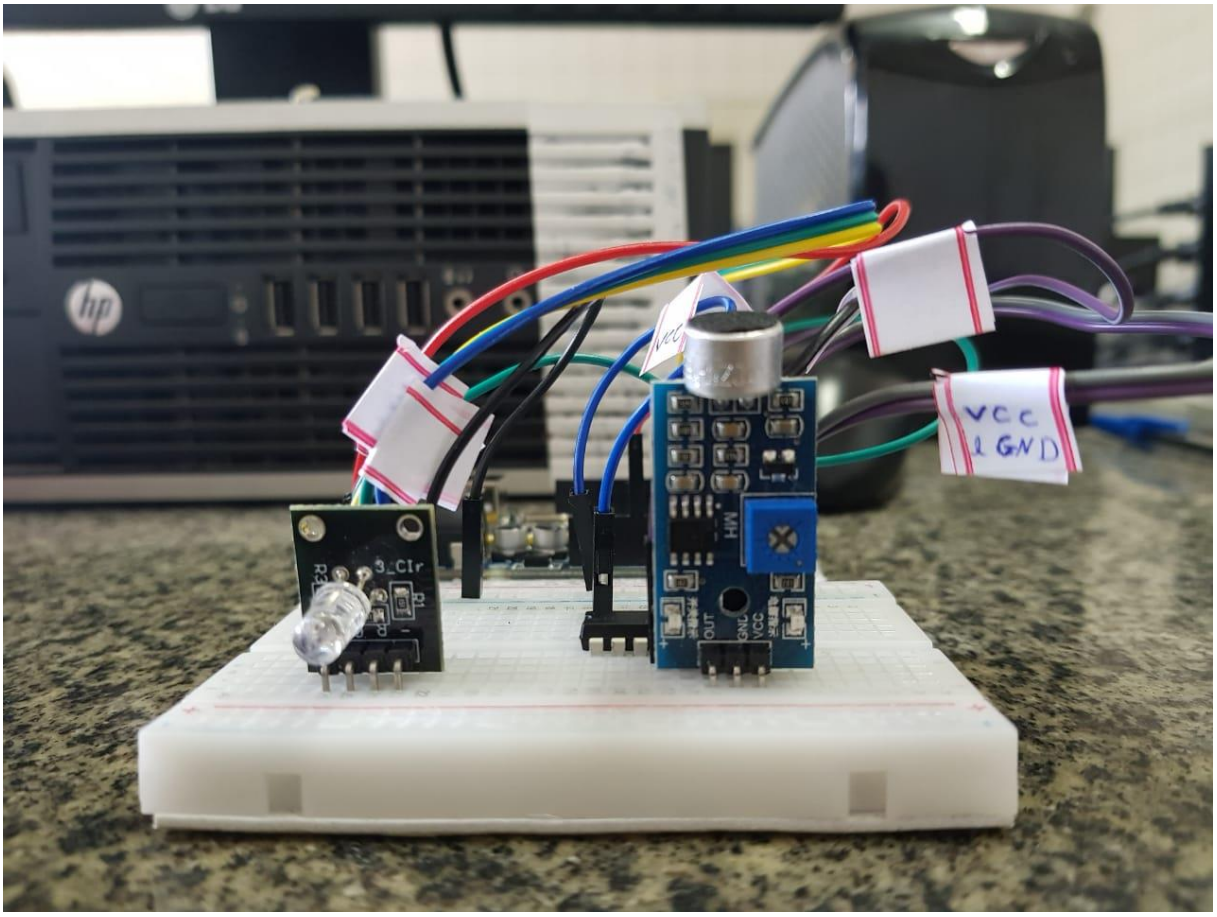
```
                        if (lowduration > (hightimesavg*(2*lacktime)) &&
lowduration < hightimesavg*(5*lacktime)){
                                funcaoDecodificar();
                                code[0] = '\0';
                                Serial.print(" ");
                        }

                        //pular linha
                        if (lowduration >= hightimesavg*(5*lacktime)){
                                funcaoDecodificar();
                                code[0] = '\0';
                                Serial.println();
                        }
                }
        }

        //passou o tempo de ter letras
        if ((millis() - startttimelow) > (highduration * 6) && stop == LOW){
                funcaoDecodificar();
                code[0] = '\0';
                stop = HIGH;
        }

        //fim do loop
        realstatebefore = realstate;
        lasthighduration = highduration;
        filteredstatebefore = filteredstate;
}



void funcaoDecodificar(){
        //Alfabeto em maiusculo na tabela ASCII
        if (strcmp(code,".-") == 0) funcaoCodificada(65);
        if (strcmp(code,"-...") == 0) funcaoCodificada(66);
        if (strcmp(code,"-.-.") == 0) funcaoCodificada(67);
        if (strcmp(code,"-..") == 0) funcaoCodificada(68);
        if (strcmp(code,".") == 0) funcaoCodificada(69);
        if (strcmp(code,"..-.") == 0) funcaoCodificada(70);
        if (strcmp(code,"--.") == 0) funcaoCodificada(71);
        if (strcmp(code,"....") == 0) funcaoCodificada(72);
        if (strcmp(code,"..") == 0) funcaoCodificada(73);
        if (strcmp(code,".---") == 0) funcaoCodificada(74);
        if (strcmp(code,"-.-") == 0) funcaoCodificada(75);
        if (strcmp(code,".-..") == 0) funcaoCodificada(76);
        if (strcmp(code,"--") == 0) funcaoCodificada(77);
        if (strcmp(code,"-.") == 0) funcaoCodificada(78);
        if (strcmp(code,"---") == 0) funcaoCodificada(79);
        if (strcmp(code,".--.") == 0) funcaoCodificada(80);
        if (strcmp(code,"--.-") == 0) funcaoCodificada(81);
        if (strcmp(code,".-.") == 0) funcaoCodificada(82);
        if (strcmp(code,"...") == 0) funcaoCodificada(83);
        if (strcmp(code,"-") == 0) funcaoCodificada(84);
        if (strcmp(code,"..-") == 0) funcaoCodificada(85);
        if (strcmp(code,"...-") == 0) funcaoCodificada(86);
        if (strcmp(code,".--") == 0) funcaoCodificada(87);
        if (strcmp(code,"-..-") == 0) funcaoCodificada(88);
        if (strcmp(code,"-.--") == 0) funcaoCodificada(89);
        if (strcmp(code,"--..") == 0) funcaoCodificada(90);
```

```
        //Codigos especiais para comandos
        if (strcmp(code,"..-...") == 0) desligarLEDS();
        if (strcmp(code,".--.-.") == 0) ligarRoxo();
}

void funcaoCodificada(int numeroASCII){
        //Serial.print(char(numeroASCII));
        soma+= numeroASCII;

        if(soma == 219){
                mexerLED();
                Serial.println("Entendido!");
                soma = 0;
        }else if(soma > 219){
                soma = 0;
        }

}
void mexerLED(){
        switch (ledAtual){
        case 0:
                ligarVermelho();
                break;
        case 1:
                ligarVerde();
                break;
        case 2:
                ligarAzul();
                break;
        default:
                ligarVermelho();
                break;
        }

}

void desligarLEDS(){
        Serial.println("Desligando...");
        digitalWrite(pinoVermelho,LOW);
        digitalWrite(pinoAzul,LOW);
        digitalWrite(pinoVerde,LOW);
        ledAtual = 0;
}

void ligarVermelho(){
        desligarLEDS();
        Serial.println("Ligando Vermelho!");
        digitalWrite(pinoVermelho,HIGH);
        ledAtual = 1;
}

void ligarVerde(){
        desligarLEDS();
        Serial.println("Ligando Verde!");
        digitalWrite(pinoVerde,HIGH);
        ledAtual = 2;
}
```

```
void ligarAzul(){
       desligarLEDS();
       Serial.println("Ligando Azul!");
       digitalWrite(pinoAzul,HIGH);
       ledAtual = 3;
}

void ligarRoxo(){
       desligarLEDS();
       Serial.println("Ligando Roxo!");
       digitalWrite(pinoVermelho,HIGH);
       digitalWrite(pinoAzul,HIGH);
       ledAtual = 4;
}
```

FOTOS




VCC
GND
OUT
GND
VCC