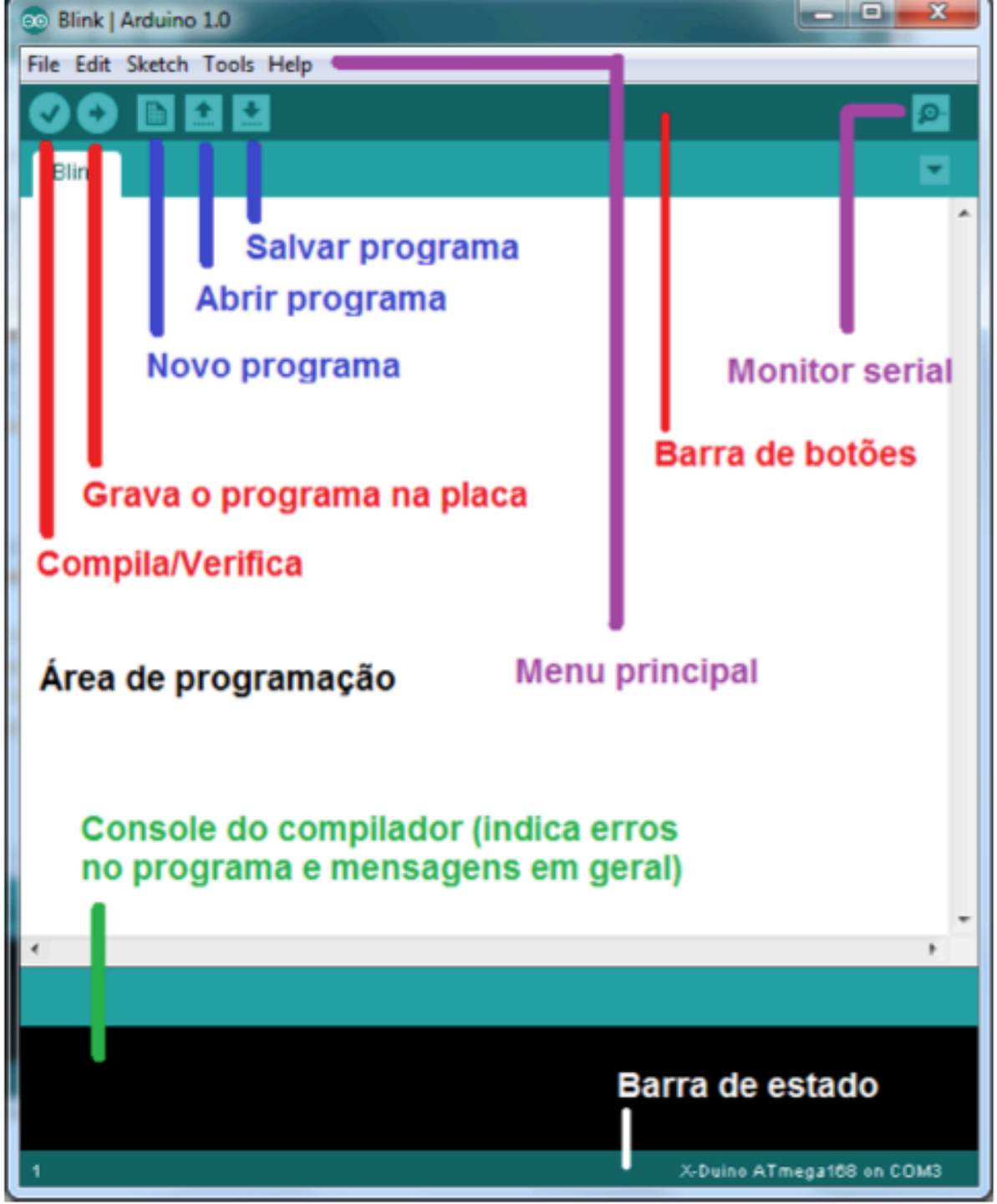
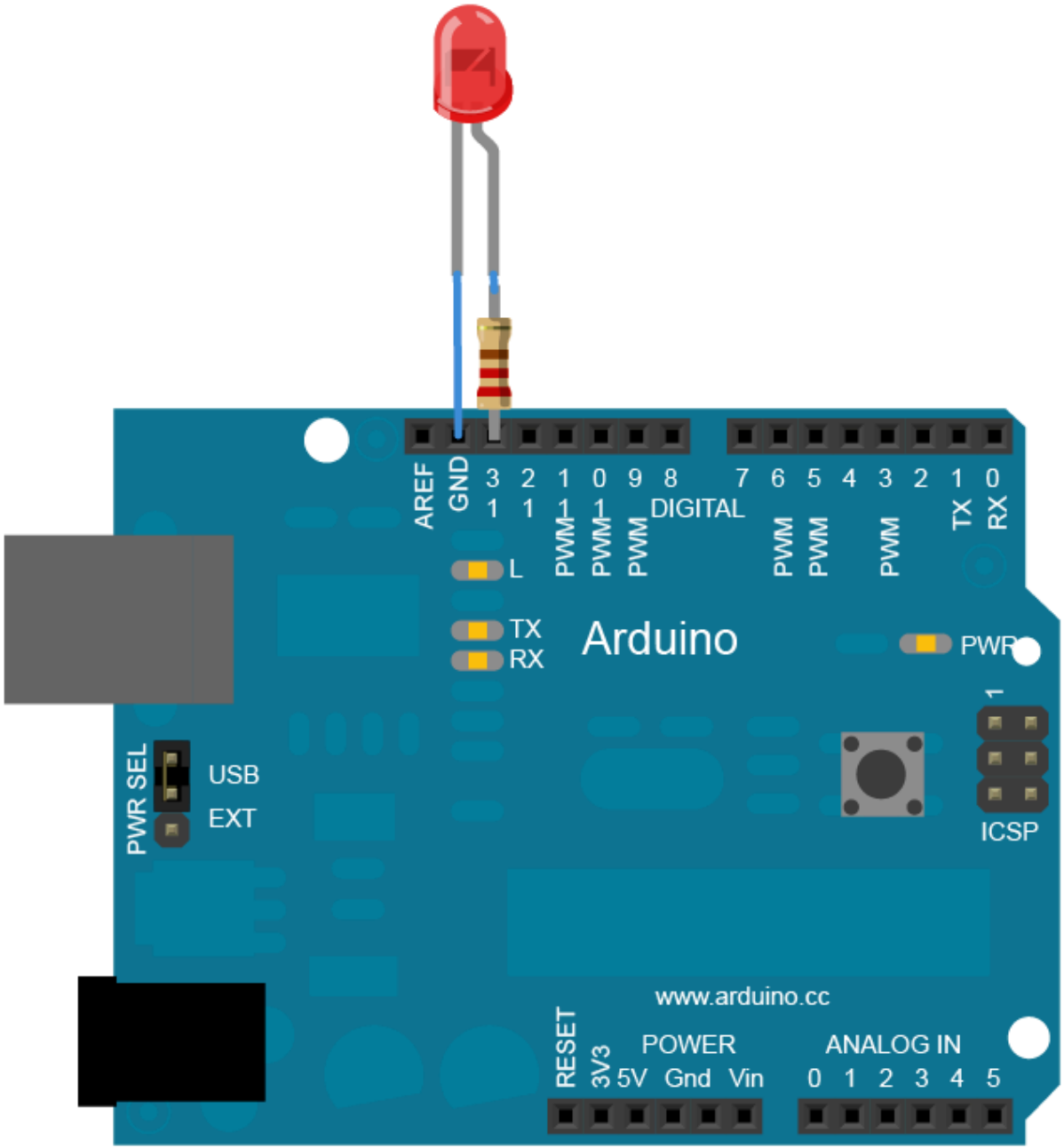
Práticas Arduino

Manoel Neto


Blink | Arduino 1.0
File Edit Sketch Tools Help
Salvar programa
Abrir programa
Novo programa
Monitor serial
Barra de botões
Grava o programa na placa
Compila/Verifica
Área de programação
Menu principal
Console do compilador (indica erros
no programa e mensagens em geral)
Barra de estado
1
X-Duino ATmega168 on COM3

Hello blink

```
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(500);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(500);               // wait for a second
}
```


AREF
GND
3
2
1
0
9
8
7
6
5
4
3
2
1
0
DIGITAL
PWM
PWM
PWM
PWM
PWM
PWM
TX
RX
L
TX
RX
Arduino
PWR
PWR SEL
USB
EXT
1
ICSP
www.arduino.cc
RESET
3V3
POWER
5V
Gnd
Vin
ANALOG IN
0
1
2
3
4
5

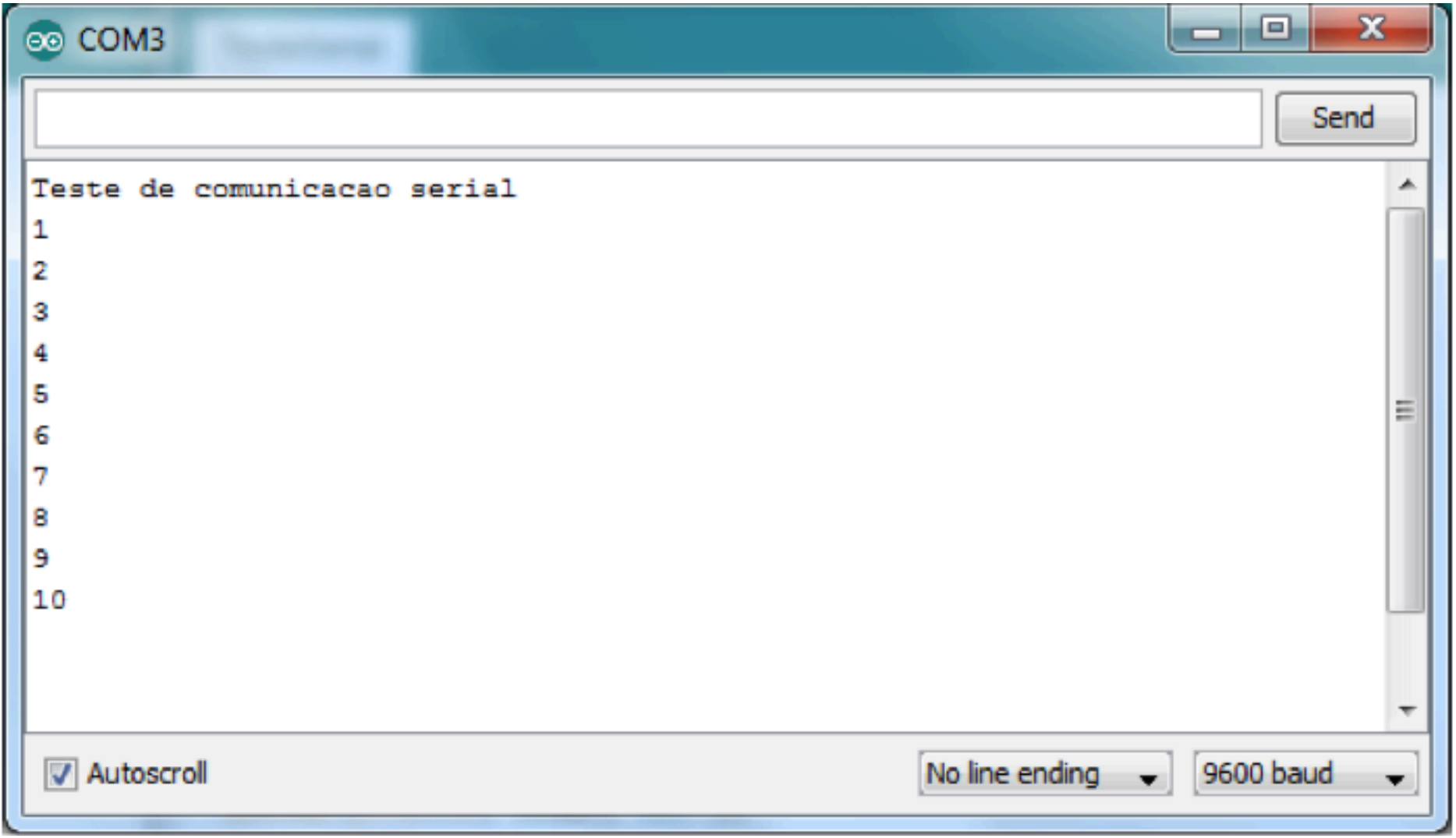
Hello Blink ”s”

- Repita o exercício anterior para 3 leds. Use um vetor de inteiros para definir os pinos utilizados.

Monitor Serial

- Monitor usado para que possamos comunicar nossa placa com o computador.
- Muito útil para a depuração do programa.
- Basicamente conectamos a placa no computador e através da tela podemos ver as informações enviadas pela placa.


File Edit Sketch Tools Help
Serial Monitor
TesteSerial §
/*
Teste Serial
*/
//Função de inicialização
void setup() {
Serial.begin(9600); //inicializa a comunicação na velocidade 9600
Serial.println("Teste de comunicacao serial"); //envia texto
}
int count = 0;
//looping principal
void loop() {
count++;
Serial.println(count, DEC); //envia o valor do contador
delay(1000); // espera um segundo (1000 mS)
}


COM3
Send
Teste de comunicacao serial
1
2
3
4
5
6
7
8
9
10
Autoscroll
No line ending
9600 baud

Funções Importantes

- ***pinMode (pin, mode)***: Configura o pino especificado para que se comporte como entrada ou saída, sendo Pin = número do pino e mode = INPUT ou OUTPUT
- ***digitalWrite (pin,value)***: escreve um valor HIGH ou LOW em um pino digital.
  - Se o pino foi configurado como saída sua voltagem será determinada ao valor correspondente: 5V para HIGH e 0V para LOW.
  - Se o pino estiver configurado como entrada escrever um HIGH levantará o resistor interno de 20kΩ. Escrever um LOW rebaixará o resistor.

Funções Importantes

- ***int digitalRead (pin)***: Lê o valor de um pino digital especificado, HIGH ou LOW. Pin = numero do pino. Retorna HIGH ou LOW.
- ***int analogRead (pin)***: Lê o valor de um pino analógico especificado. Pode mapear voltagens entre 0 a 5v, sendo 4,9mV por unidade.

Funções Importantes

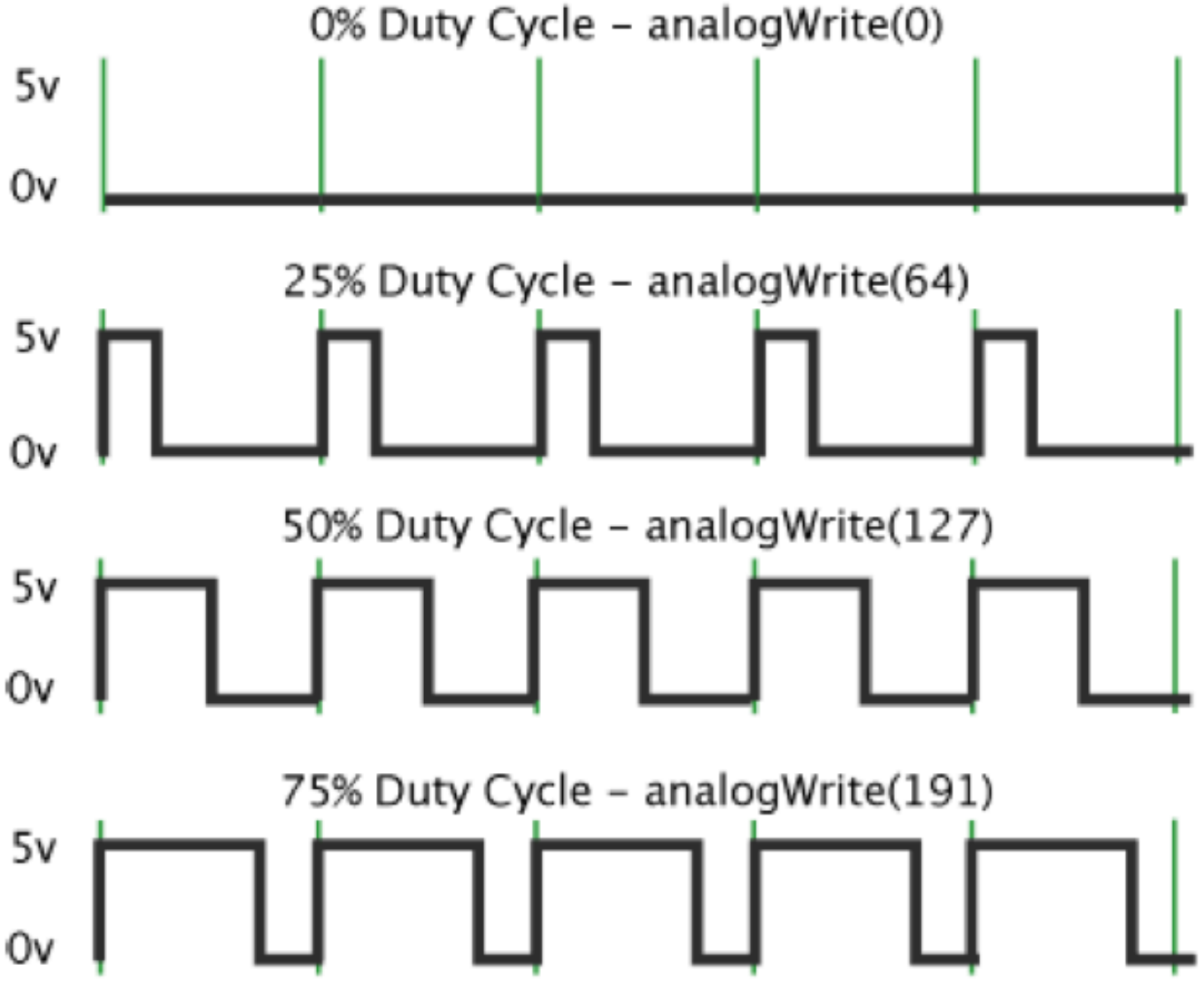
- ***analogWrite (pin, value)***: Escreve um valor analógico. Pode ser utilizada para acender um LED variando o brilho ou girar um motor a velocidade variável.
- PWM significa modulação por largura de pulso (***Pulse Width Modulation***) e é basicamente uma técnica para obtermos resultados analógicos em meios digitais.

```
#define LED 11

void setup (){
  pinMode(LED, OUTPUT); //pino 11 ajustado como saída
}

void loop (){
  int i;
  for (i=0; i<255; i++){// variando i de 0 a 2255
    analogWrite(LED,i);// escrevendo o valor de i no pino 11
    delay(30);// esperando 30 milésimos de segundo
  }
}
```

PWM

- A função analogWrite (), apesar de estarmos utilizando uma porta digital, é a responsável pelo PWM e recebe como parâmetro o pino e um valor entre 0 – 255, em que o 0 corresponde a 0% e 255 corresponde a 100% do *duty cycle*

Duty Cicle


0% Duty Cycle – analogWrite(0)
5v
0v
25% Duty Cycle – analogWrite(64)
5v
0v
50% Duty Cycle – analogWrite(127)
5v
0v
75% Duty Cycle – analogWrite(191)
5v
0v

Ultrassom

- Arduino possui muitos sensores
- Vamos aprender a controlar o Ultrassom
- Um único sensor de ultrassom, possui um receptor e um emissor. Como funciona:
  - Emite um sinal na faixa de frequência do ultrassom (por volta de 30kHz)
  - sinal se propaga pelo ar até encontrar um obstáculo
  - Ao colidir com o obstáculo uma parte do sinal é refletida e captada pelo sensor .

Ultrassom

Ultrassom

- Precisaremos de dois pinos (Emissor e Receptor)
- Um como saída (que emite o sinal) e outro como entrada (que recebe o sinal)
- Pino que envia o pulso é chamado de ***trigger*** e o que recebe ***echo***

```
#define echoPin 13 //Pino 13 recebe o pulso do echo
#define trigPin 12 //Pino 12 envia o pulso para gerar o echo
void setup()
{
   Serial.begin(9600); //inicia a porta serial
   pinMode(echoPin, INPUT); // define o pino 13 como entrada (recebe)
   pinMode(trigPin, OUTPUT); // define o pino 12 como saida (envia)
}
void loop()
{
  //seta o pino 12 com um pulso baixo "LOW" ou desligado ou ainda 0
    digitalWrite(trigPin, LOW);
  // delay de 2 microssegundos
    delayMicroseconds(2);
  //seta o pino 12 com pulso alto "HIGH" ou ligado ou ainda 1
    digitalWrite(trigPin, HIGH);
  //delay de 10 microssegundos
    delayMicroseconds(10);
  //seta o pino 12 com pulso baixo novamente
    digitalWrite(trigPin, LOW);
  //pulseInt lê o tempo entre a chamada e o pino entrar em high
    long duration = pulseIn(echoPin,HIGH);
  //Esse calculo é baseado em s = v . t, lembrando que o tempo vem dobrado
  //porque é o tempo de ida e volta do ultrassom
    long distancia = duration /29 / 2 ;
Serial.print("Distancia em CM: ");
Serial.println(distancia);
delay(1000); //espera 1 segundo para fazer a leitura novamente
}
```

Ultrassom

- int pingPin = 13;
- int inPin = 12;

- void setup() {
- pinMode(pingPin, OUTPUT);
- pinMode(inPin, INPUT);
- Serial.begin(9600);
- }
- // The same pin is used to read the signal from the PING))): a HIGH
- // pulse whose duration is the time (in microseconds) from the sending
- // of the ping to the reception of its echo off of an object.
-
- duration = pulseIn(inPin, HIGH);

- Continua....

Ultrassom

```
  // convert the time into a distance
  inches = microsecondsToInches(duration);
  cm = microsecondsToCentimeters(duration);


  Serial.print(inches);
  Serial.print("in, ");
  Serial.print(cm);
  Serial.print("cm \n");


  delay(100);
}
```

Ultrassom

- **long microsecondsToInches(long microseconds)**
- **{**
  - **return microseconds / 74 / 2;**
- **}**
- **long microsecondsToCentimeters(long microseconds)**
  - **{**

    return microseconds / 29 / 2;
  - **}**

Blink Sound

- Acenda um Led a cada vez que um um obstáculo chegar a menos que 15 cm do ultrassom.

Alarme

- Faça um alarme tocar se a distancia de sua mão for maior que 15 cm. Use um ultrassom, uma buzina (buzzer) e três leds coloridos.
  - const int Buzzer = 6;
  - const int led1 = 8;
  - const int led2 = 9;
  - const int led2 = 10;

Alarme de Incêndio

- Ler o valor do sensor com analogRead(PinoSensor);
- Monte uma escala de valores e acenda os leds de acordo com esta escala(proximo de 15, meio longe de 15 e muito longe de 15).
- Quando os três leds forem acesos ligue a buzina.

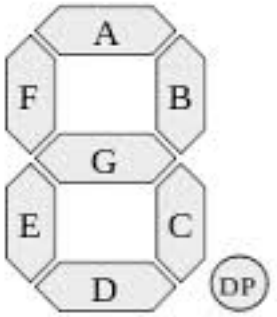
Desafio!

- Escreva de 0 a 9 usando um display de 7 segmentos.
- Depois use 2 displays de 7 segmentos e escreva até 99.

DSP 7 Seg

DSP 7 Seg

- O **display de 7 segmentos**, como o próprio nome diz, tem 7 partes, ou segmentos, que podem ser agrupados de modo a formar números e letras. Os segmentos são organizados de A a F.



DSP 7 Seg

- Assim, se você quiser mostrar o número 1, basta ativar os segmentos B e C. Para mostrar o número 3, os segmentos A, B, C, D e G, e assim por diante.

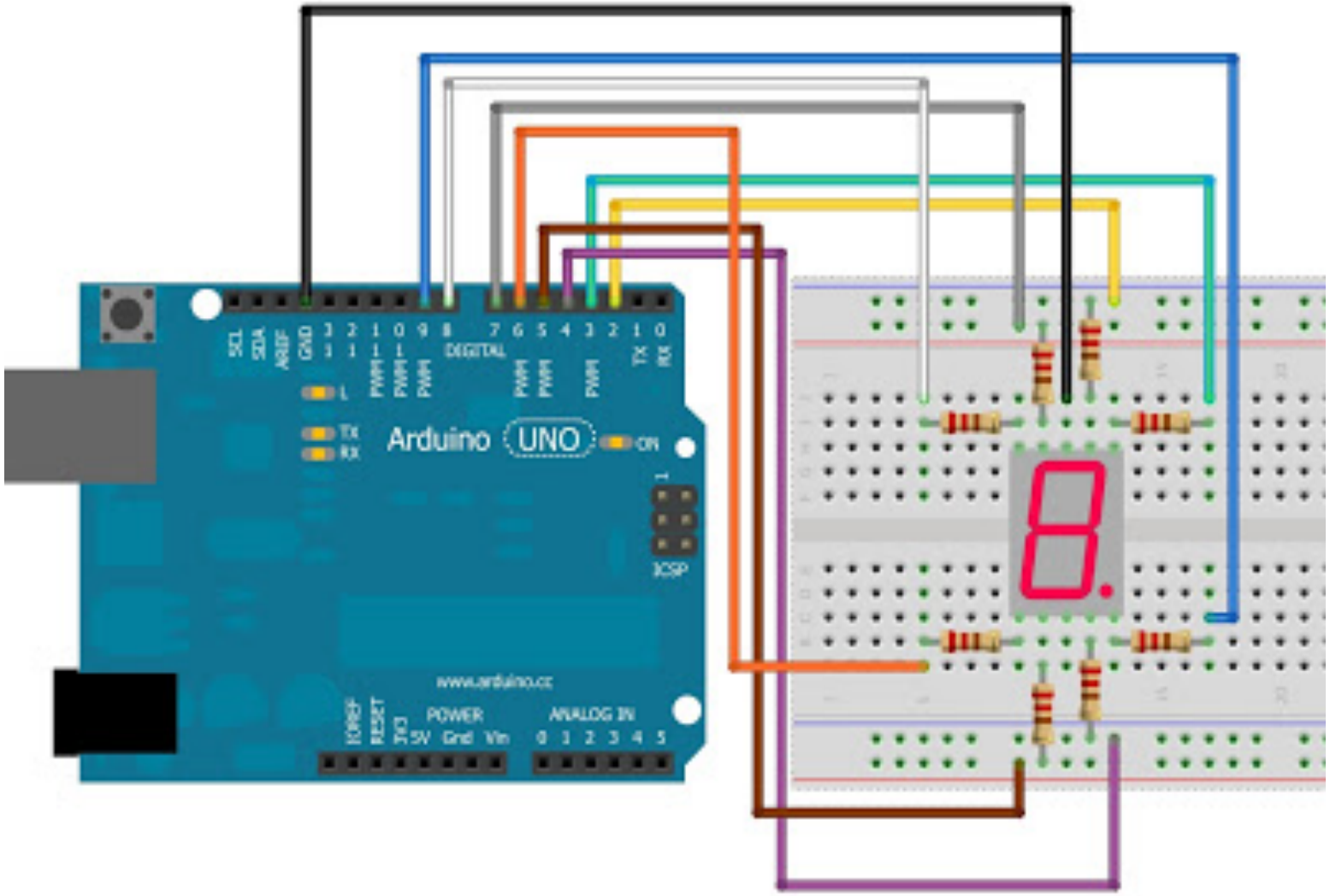
DSP 7 Seg

- A ligação ao Arduino usa os pinos de 2 a 9, mais o GND (ou VCC se seu modelo for Anodo Comum), na seguinte sequencia :
  - **Pino 2 do Arduino ====> Pino segmento A**
  - **Pino 3 do Arduino ====> Pino segmento B**
  - **Pino 4 do Arduino ====> Pino segmento C**
  - **Pino 5 do Arduino ====> Pino segmento D**
  - **Pino 6 do Arduino ====> Pino segmento E**
  - **Pino 7 do Arduino ====> Pino segmento F**
  - **Pino 8 do Arduino ====> Pino segmento G**
  - **Pino 9 do Arduino ====> Pino segmento PONTO**
  - **Pino GND do Arduino => Pino 3 do display**

DSP 7 Seg

- Um ponto importante é a utilização dos resistores de 220 ohms para cada pino. Como o display trabalha com 2V, é necessário limitar a corrente, evitando queimar o componente :

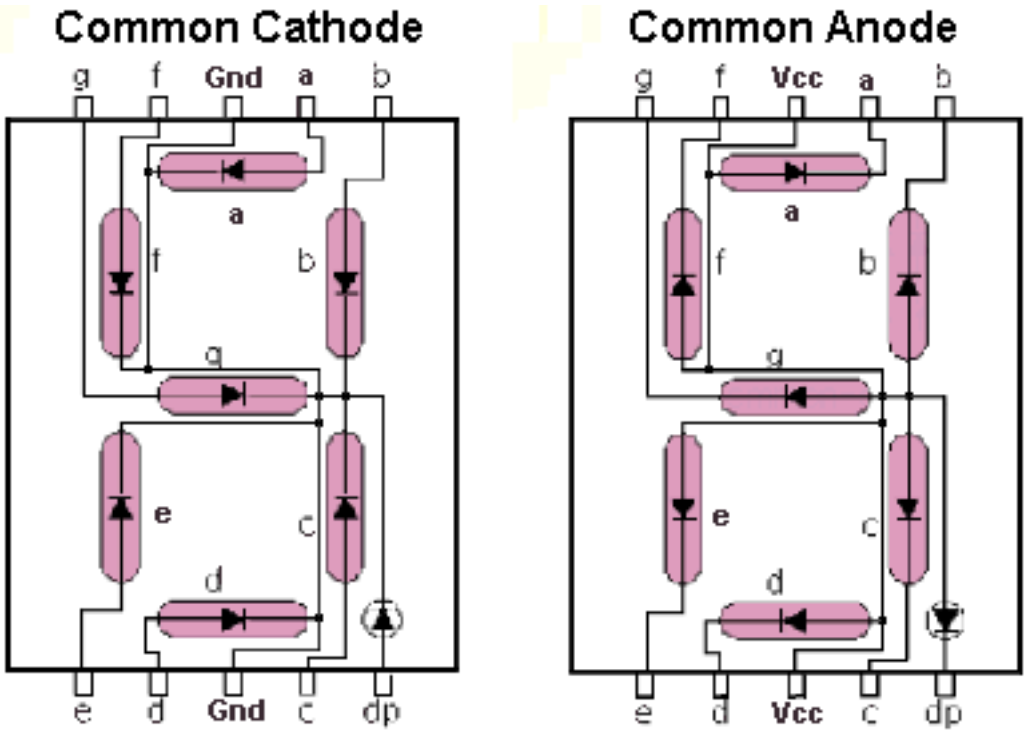
DSP 7 Seg



DSP 7 Seg

- A disposição dos pinos é mostrada na próxima imagem. Se o seu display for de outro modelo, basta descobrir qual pino corresponde
- a cada segmento. Isso pode ser feito utilizando-se um multimetro ou até mesmo uma pilha, tomando o cuidado de respeitar a voltagem máxima que comentei acima : 2 volts.

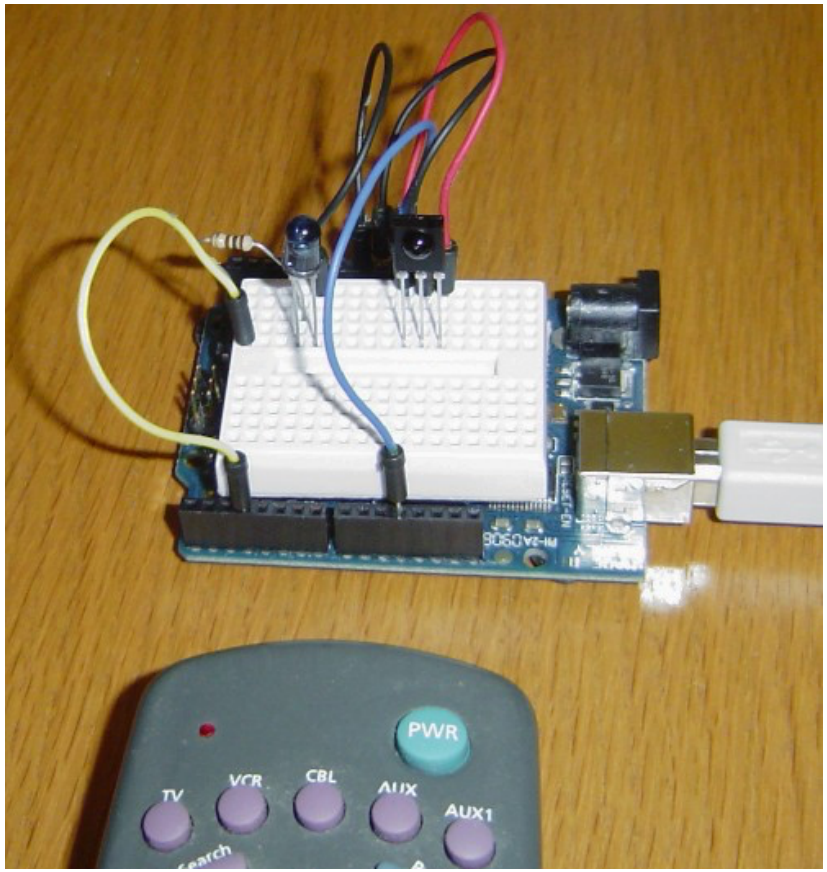
DSP 7 Seg


Common Cathode
g f Gnd a b
a
f b
g
e c
d
e d Gnd c dp
Common Anode
g f Vcc a b
a
f b
g
e c
d
e d Vcc c dp

Desafio Matriz de Leds

- Use uma Matriz de leds 8x8 para imprimir a mensagem “Eu sou aluno do GSORT ☺ ”… incluido a carinha feliz !!!!

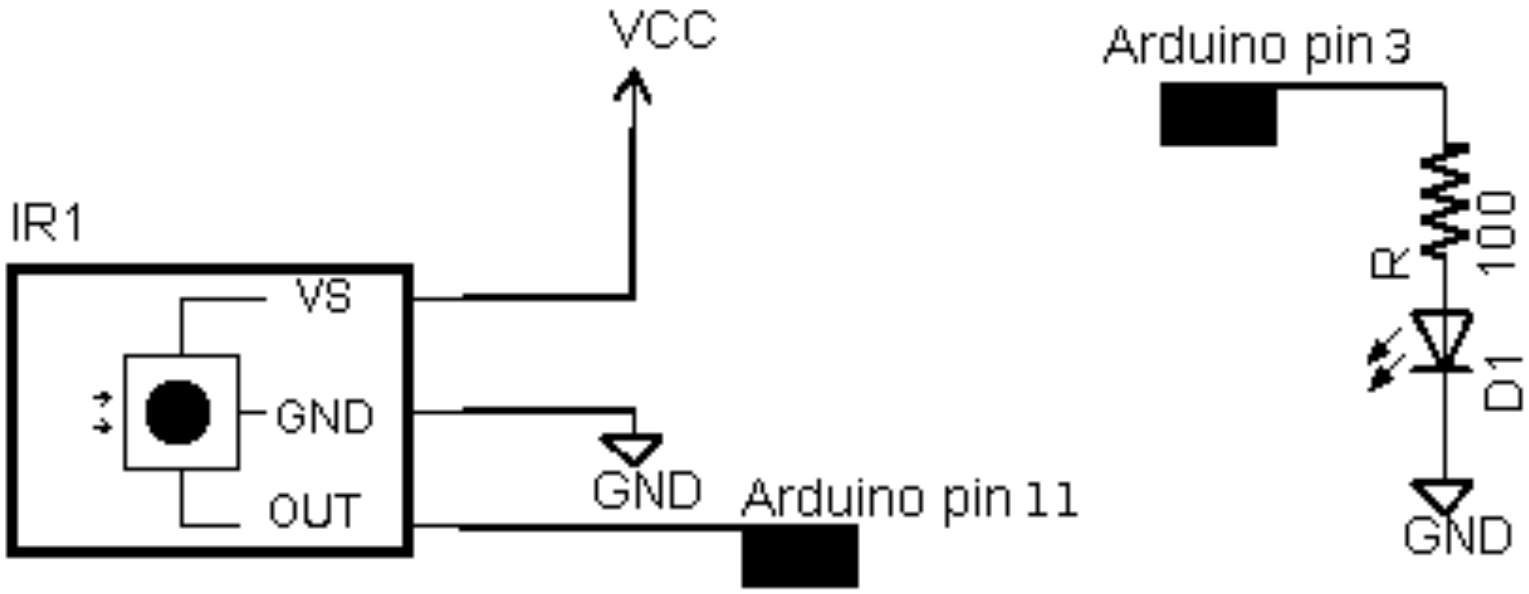
InfraRed



InfraRed

1. Baixar e instalar a IRRemote https://www.pjrc.com/teensy/arduino_libraries/IRremote.zip
2. Usar um sensor para recepÇão e um led (IR) para emissão.
3. Mais detalhes em https://www.pjrc.com/teensy/td_libs_IRremote.html

InfraRed



Exemplo de Recepção:

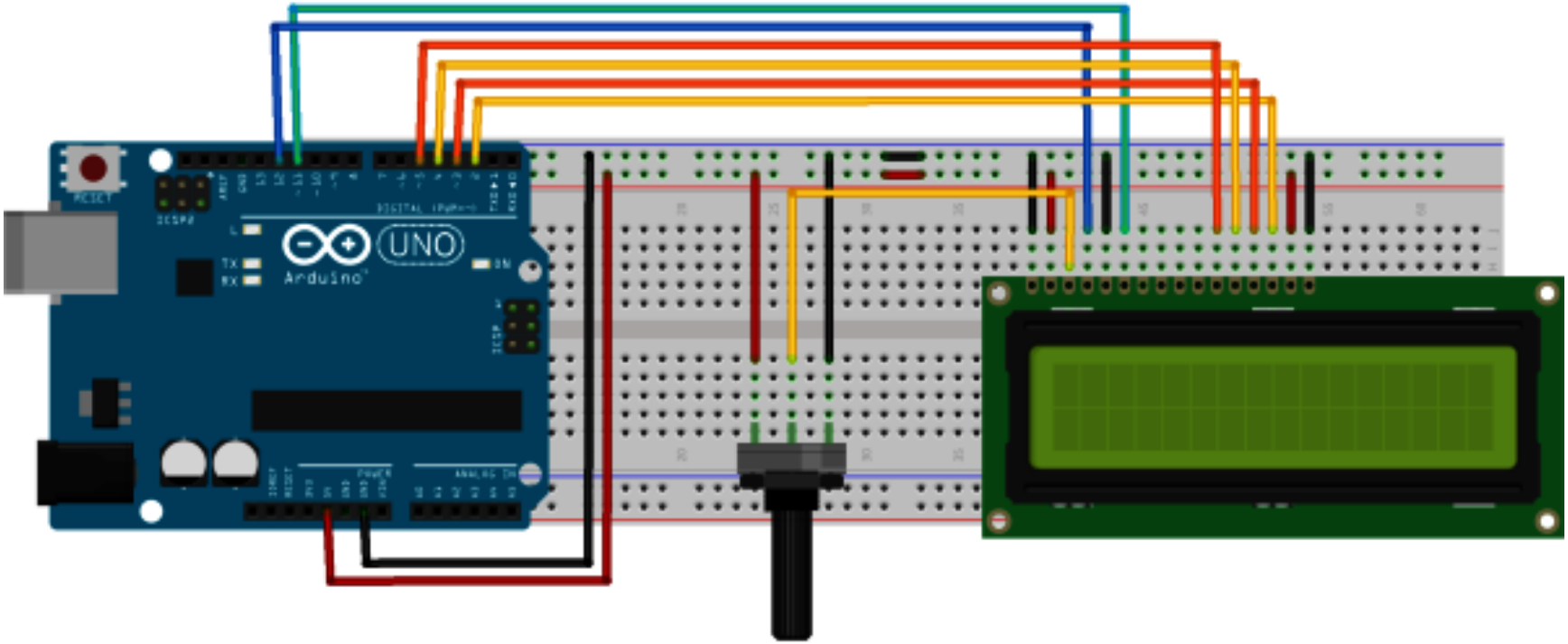
```
#include <IRremote.h>

const int RECV_PIN = 6;

IRrecv irrecv(RECV_PIN);

decode_results results;

void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // Start the receiver
  irrecv.blink13(true);
}

void loop() {
  if (irrecv.decode(&results)) {
    if (results.decode_type == NEC) {
      Serial.print("NEC: ");
    } else if (results.decode_type == SONY) {
      Serial.print("SONY: ");
    } else if (results.decode_type == RC5) {
      Serial.print("RC5: ");
    } else if (results.decode_type == RC6) {
      Serial.print("RC6: ");
    } else if (results.decode_type == UNKNOWN) {
      Serial.print("UNKNOWN: ");
    }
    Serial.println(results.value, HEX);
    irrecv.resume(); // Receive the next value
  }
}
```

Exemplo de Emissão:

```
#include <IRremote.h>

IRsend irsend;

void setup() {
}

void loop() {
    irsend.sendSony(0x68B92, 20);
    delay(100);
    irsend.sendSony(0x68B92, 20);
    delay(100);
    irsend.sendSony(0x68B92, 20);
    delay(300000);
}
```

Display LCD

Pino	Símbolo	Função
1	VSS	GND(Alimentação)
2	VDD	5V(Alimentação)
3	V0	Ajuste de Contraste
4	RS	Habilida/Desabilita Seletor de Registrador
5	R/W	Leitura/Escrita
6	E	Habilita Escrita no LCD
7	DB0	Dado
8	DB1	Dado
9	DB2	Dado
10	DB3	Dado
11	DB4	Dado
12	DB5	Dado
13	DB6	Dado
14	DB7	Dado
15	A	5V(Backlight)
16	K	GND(BackLight)


UNO
Arduino

Exemplo LCD

- #include <LiquidCrystal.h> //Inclui a biblioteca do LCD
-
- LiquidCrystal lcd(12, 11, 5, 4, 3, 2); //Configura os pinos do Arduino para se comunicar com o LCD
-
- int temp; //Inicia uma variável inteira(temp), para escrever no LCD a contagem do tempo
-
- void **setup**()
- {

- lcd.begin(16, 2); //Inicia o LCD com dimensões 16x2(Colunas x Linhas)
- lcd.setCursor(0, 0); //Posiciona o cursor na primeira coluna(0) e na primeira linha(0) do LCD
- lcd.print("Ola Mundo!"); //Escreve no LCD "Olá Mundo!"
- lcd.setCursor(0, 1); //Posiciona o cursor na primeira coluna(0) e na segunda linha(1) do LCD
- lcd.print(”GSORT"); //Escreve no LCD ”GSORT"
-
- }
- Continua….

Exemplo LCD

- void **loop**()
- {

- lcd.setCursor(13, 1); //Posiciona o cursor na décima quarta coluna(13) e na segunda linha(1) do LCD
- lcd.print(temp); //Escreve o valor atual da variável de contagem no LCD
- delay(1000); //Aguarda 1 segundo
- temp++; //Incrementa variável de contagem
-
- if(temp == 600) //Se a variável temp chegar em 600(10 Minutos),...
- {
- temp = 0; //...zera a variável de contagem
- }

- }