**Programming the Physical World**

**Morse Code Translator**

In this lab we are going to write a program that interfaces with computer hardware to make something physical occur, specifically blink Morse code on a Light Emitting Diode (LED).

For this activity we will require the following:

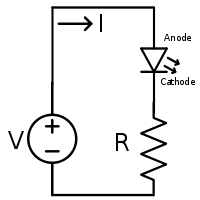
1. Arduino Uno
2. Two male/male jumper wires
3. Solderless breadboard
4. LED
5. 220 Ω resistor

The hardware we will be using will be the Arduino Uno which has Pin 8 (Input/Output) pin which can be easily programmed to send output signals to or receive input signals.

A screenshot of a computer

Description automatically generated

For this activity, we will be using pins 8 and 14. These correspond to Ground and 5V respectively. Here is a diagram of an LED setup. To get the LED to light we need approximately 2.1 volts across it. The purpose of the resistor is to limit the current through the LED. Pin 8 is capable of supplying 5 volts. If we do not use a current limiting resistor, the LED will light brightly for a while and then go dark. For a nominal brightness we want about 10 mA to flow through the LED. This means a 220 Ω resistor should do. Actually we should be OK with a resistor or 180 Ω to 550 Ω.



We will be using a solderless breadboard to hook up the components. Pin 6 (Ground) should hook to one end of the resistor. Pin 12 (GPIO 18) should hook to the Anode of the LED (the longer lead).

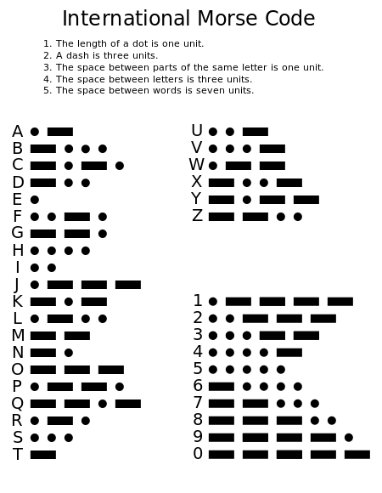
**Hardware Assembly Procedure**

1. Add 220 Ω resistor to the breadboard.
2. Add LED to breadboard so Cathode of LED (shorter lead) is connected to resistor.
3. Connect Pin 14 (Ground) on Arduino to unconnected lead of resistor.
4. Connect Pin 8 on Arduino to Anode of LED (longer lead).

Note: If you have the LED hooked up backwards it will not light and you can correct this by reversing its connections.

Now we are ready to program the Arduino. To do this, we will write a short script and connect the Arduino to USB. Open the Arduino IDE and add the code. Make sure you have selected the correct COM port.

Note: Make a folder on your desktop called myArduino for storing your programs.



Here is the code to make this happen.

//Morse Code Generator

//Author: nmessa

//Date: 6.6.2022

String phrase;

void setup() {

pinMode(8, OUTPUT);

Serial.begin(9600);

}

void loop() {

Serial.println("Enter a phrase: ");

while (Serial.available() == 0){} //wait for input

phrase = Serial.readString();

codeGen(phrase);

}

void codeGen(String s)

{

String output = "";

s.toUpperCase();

for (int i = 0; i < s.length(); i++)

{

if(s[i] == '\n')

{

output += "\n";

continue;

}

output += convert(s[i]);

output += " ";

}

Serial.println(output);

for (int i = 0; i < output.length(); i++)

{

if (output[i] == '.')

dot();

if (output[i] == '-')

dash();

if (output[i] == ' ')

delay(100);

}

}

String convert(char letter)

{

String table[26] = {".-", "-...", "-.-.", "-..", ".", "..-.", "--.", "....", "..", ".---", "-.-",

".-..", "--", "-.", "---", ".--.", "--.-", ".-.", "...", "-", "..-", "...-", ".--", "-..-",

"-.--", "--.."};

return table[letter - 65];

}

void dot()

{

digitalWrite(8, HIGH);

delay(100);

digitalWrite(8, LOW);

delay(100);

}

void dash()

{

digitalWrite(8, HIGH);

delay(1000);

digitalWrite(8, LOW);

delay(100);

}

Notes on codeGen function:

Build the output string sending each character in s to the convert function and getting the Morse code for that character. Put three spaces between each character to be consistent with the International Morse Code Standard. When s has been converted, print the result to the screen.

Display each character by flashing LED in the Morse Code string. If the character is '.', call dot(). If the character is a '-', call dash(). If the character is a space ' ', delay(100) 100 milliseconds.