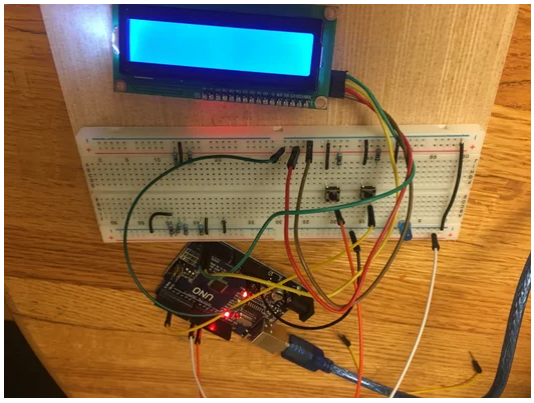
AUTODESK Instructables

Arduino Morse Code

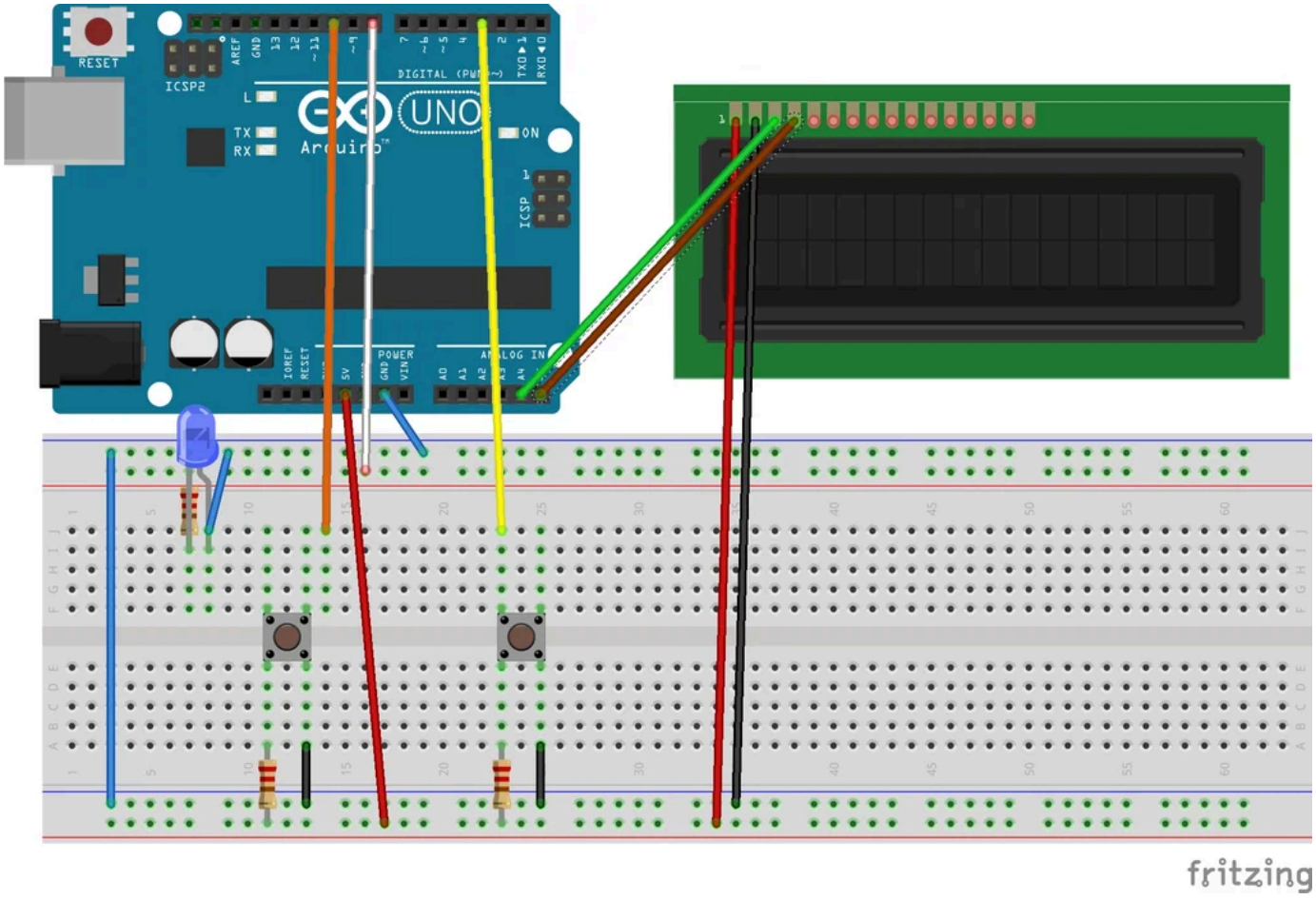
By jfk2bd in CircuitsArduino

Introduction: Arduino Morse Code



I made a Morse-to-English and a English-to-Morse encoder and decoder. This entails the use of two buttons, the LCD display, an LED, and an Arduino Uno. The functionality is as follows: one can enter a Morse code message by using the two buttons (dot/dash and go) and the Arduino chip will translate that to English and show it on the LCD/serial monitor. One can also enter a English phrase on the serial monitor and have the LED light up with the corresponding Morse code flashes.

Step 1: Electronic Design



The electronic design is fairly simple.

The LED is connected to pin 8 through a 220 ohm resistor. The two buttons are connected to a 5 volt rail through a 220 ohm resistor on one side and a pin input (3 and 10 respectively) on the other side.

The LCD display wasn't actually connected through its base pins, instead I used a backpack with 4 pins (5v, ground, SDA connected to the A4 pin, and SCL connected to the A5 pin).

Step 2: Setting Up the Code

```
#include <LiquidCrystal_I2C.h>

#include  <Wire.h>

int led = 8;
int button = 3;
int  transmit = 10;
int go = 0;
int len =0;
String fin = "";
String morse_array[50];
int test = 0;
int m_inc = 0;
char m_in[5];
String translated ="";
String morsecode[] ={
    ".-", "-...", "-.-.", "-..", ".", "..-.", "--.", "....",
    "..", ".---", "-.-", ".-..", "--", "-.", "---", ".--.",
    "--.-", ".-.", "...", "-", "..-", "...-", ".--", "-..-",
    "-.--", "--..", " "
};
String alpha [] = {"a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p",
"q", "r", "s", "t", "u", "v", "w", "x", "y", "z", " "};

LiquidCrystal_I2C lcd(0x3F, 16, 2);

void setup() {
  pinMode(led, OUTPUT);
  pinMode(button, INPUT);
  pinMode(transmit, INPUT);
  lcd.begin();Serial.begin (9600);
}
```

The first thing I do is import the relevant libraries for the LCD. I then set the pin numbers for the buttons and LED. Next I instantiate some global variables that will be used later on. I also create two dictionaries for future reference, an alphabet and a morse alphabet.

Then I construct the I2C display with the relevant address.

In the setup phase, I set the LED as output and the buttons as inputs. I also begin the LCD and the serial monitor.

Step 3: The Loop

```
void loop() {

if (Serial.available() > 0) {
  String ad = Serial.readString(); // read the incoming byte:
  char charBuf[50];
  ad.toCharArray(charBuf, 50) ;
  len = ad.length();
  convert_to_morse(charBuf);
  Serial.println(morse_array[0]);
  output_morse(morse_array);
  }
  lcd.write(fin);
if  (digitalRead(transmit)==LOW)
  {
    delay (300);
    if  (digitalRead(transmit)==HIGH)
      convert();
    else
    while (digitalRead(transmit)==LOW)
      check_button();
  }
}
```

Within the loop, there are two distinct phases. One checks if the Serial monitor is being used (implying someone wants English to Morse translated. It converts the string input to a character array because Morse code is translated letter by letter. Then it converts the char array to an array of morse strings and outputs it to the LED.

The second part addresses a Morse to English translation. The workaround I used was to push on a "go" button in order to start transmission. Once it was depressed it started reading in the button presses. Once it was depressed again, it sent that to a converter for output.

Step 4: Decoding Morse to English

```
void check_button(){

if (digitalRead(button)==LOW)
 {
 delay (300);
  if(digitalRead(button)==HIGH)
    {
      m_in[m_inc] = '.';
      m_inc++;
      test = 1;
    }
  else
    {
       m_in[m_inc] = '-';
      m_inc++;
    test = 2;
    delay(400);
    }
 }
}
void convert(){
  for (int i = 0; i<5; i++)
    {
    translated += m_in[i];
    }
    translated.trim();
     Serial.print(translated);
    translated = findMorse(translated);
     Serial.print(translated);
     fin += translated;
    translated = "";
    for (int i = 0; i<5; i++)
    {
    m_in[i] = ' ';
    }
    m_inc=0;
}
String findMorse(String s){
  int v = 26;
  for (int x = 0; x<26;x++)
  {
    if (morsecode[x].equals(s))
      v = x;
  }

  return alpha[v];
}
```

This code is the bottom level of converting Morse to English. First it identifies whether the button push was long or short (dot or dash) and pushes the relevant character into an array. Then at the end of the message, I concatenate the characters into one string and match it up with the correct letter. I then concatenate that letter to a global string and show it on the Serial monitor and the LCD.

Step 5: Encoding English to Morse

```
void output_morse(String m[]){

  int i = 0;
  int l = 0;
  char chars[5];
  while(m[i]!="")
   {
    m[i].toCharArray(chars, 5);
    l = m[i].length();
    int t =0;
    while(t<l){
      if(chars[t]=='.'){
        output_dot();
      }
      if(chars[t]=='-'){
        output_dash();
      }
      if(chars[t]==' '){
        output_space();
      }
      t++;
    }
    i++;
   }
}
void output_dot(){
  digitalWrite(led, HIGH);
  delay(200);
  digitalWrite(led, LOW);
  delay(500);
}
void output_dash(){
  digitalWrite(led, HIGH);
  delay(1111);
  digitalWrite(led, LOW);
  delay(500);

}
void output_space(){
  delay(500);
}
<br>

void convert_to_morse(char letters[]){ <br>  for (int i = 0; i<len; i++)
    {     morse_array[i] = findLetter(letters[i]);

    }
}

String findLetter(char in){ int m =0; switch(in)

{   case 'A':     m=0;   break;   case 'B':     m=1;         break;   case 'C':     m=2;       break;   case 'D':      m=3;       break;   case 'E':

 return morsecode[m];

}
```

This is the other side of the encoder/decoder. The overall idea is to break down the entry into its character components and find the relevant morse code for each (which is what convert_to_morse does). findLetter just looks up each letter and pushes the morse code. We aren't done though. We take the array of morse codes and break each one down into a dot and dash set, which we then individually send the LED to show the message in morse, with a dash lighting up for much longer than a dot.

Step 6: Final Thoughts

I initially wanted to do this with one button but without parallel computing it was near impossible to decide between a pause between symbols and the end of the letter. I also wanted to do this while using under 50% of memory which I accomplished (49%)!

There's a lot of room for expansion for this idea, which I could potentially tackle in the future. Other than that, here's some film of it in action!