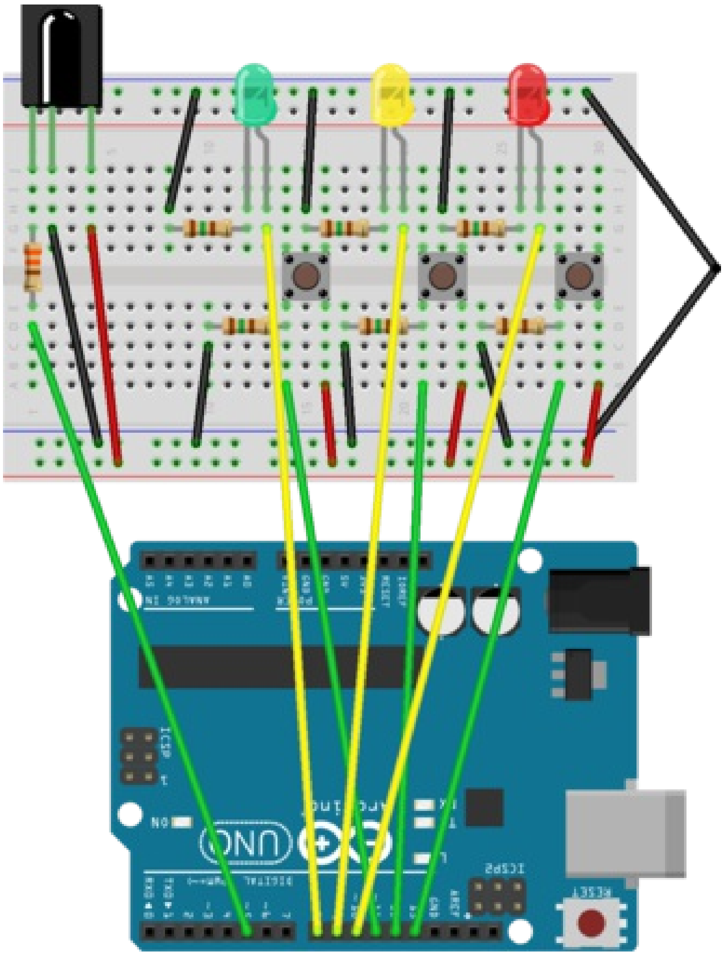
Infrared Project

This project will give you a basic understanding of how infrared light is used to communicate between devices.

Items needed:

- 1 330 Ω resistor
- 6 150 Ω resistors
- 3 LEDs
- 3 buttons
- 1 IR receiver

Hardware:



External libraries:

When developing software, often times someone else has already developed code that does what you want your code to do and has published it as a library. In our case, someone has published a library called IRremote that simplifies sending and receiving infrared signals.

Obtaining the IRremote library

1. Download a zip file containing the IRremote library at https://github.com/shirriff/Arduino-IRremote/archive/master.zip (https://github.com/shirriff/Arduino-IRremote/archive/master.zip)
2. Rename the zip file to IRremote.zip
3. In the Arduino Sketch Editor, click on Sketch->Import Library->Add Library…
4. Navigate to and select IRremote.zip
5. The IRremote library should now be available for usage in the Arduino Sketch Editor.

Code:

```
#include <IRremote.h> // Include the IRremote library installed earlier

// Create constants for all of our input/output pins
const int recButton1 = 13;
const int recButton2 = 12;
const int recButton3 = 11;
const int ledPin1 = 10;
const int ledPin2 = 9;
const int ledPin3 = 8;

// These are objects used by the IRremote library. irrecv is used to receive signals from the IR sensor and then the actual signals are decoded and placed in results.
IRrecv irrecv(5);
decode_results results;

// IR signals are not sent constantly from remotes, but instead on constant intervals. We are going to compensate for this by lighting the LEDs if an appropriate IR signal was received in the last 150 milliseconds. To do this, we need to store the time in milliseconds that the last IR signal occurred.
unsigned long lastLed1 = 0;
unsigned long lastLed2 = 0;
unsigned long lastLed3 = 0;

// Because the IRremote library returns an object of type decode_results for each IR signal received, we can store this object for each button/LED to program the LED.
decode_results ir1;
decode_results ir2;
decode_results ir3;

// NEC remotes send a useless repeat signal when a button on the remote is held down. To compensate for this, we save the last useful NEC signal and replace the repeat signal with this.
decode_results lastNEC;

void setup() {
  // Set the pin modes for all of our pins
  pinMode(recButton1, INPUT);
  pinMode(recButton2, INPUT);
  pinMode(recButton3, INPUT);
  pinMode(ledPin1, OUTPUT);
  pinMode(ledPin2, OUTPUT);
  pinMode(ledPin3, OUTPUT);

  // Clear all LED programming on launch
  ir1.decode_type = UNKNOWN;
  ir2.decode_type = UNKNOWN;
  ir3.decode_type = UNKNOWN;

  irrecv.enableIRIn();

  // Open a serial connection for debugging
  Serial.begin(9600);
}

void loop() {
  // If the IRremote library has received a signal, store it in results and continue.
  if (irrecv.decode(&results)) {
    Serial.print(results.value, HEX);
    Serial.print(" (");
    Serial.print(results.bits);
    Serial.print(" bits, ");
    switch (results.decode_type) {
      case UNKNOWN:
        Serial.print("UNKNOWN");
        break;
      case NEC:
        Serial.print("NEC");
        break;
      case SONY:
        Serial.print("SONY");
        break;
      case RC5:
        Serial.print("RC5");
        break;
      case RC6:
        Serial.print("RC6");
        break;
    }
    Serial.println(")");

    // Allow the IRremote library to continue receiving IR signals
    irrecv.resume();

    // Check if we are receiving the NEC repeat signal. If not, set lastNEC to the current IR signal. If so, use the last IR signal in the rest of the loop.
    if(results.decode_type == NEC) {
      if(results.value == 0xFFFFFFFF && results.bits == 0) {
        results = lastNEC;
      }
      else {
        lastNEC = results;
      }
    }

    // Check if the IR signal we are receiving is valid. Typically an unknown decode type indicates
that the IR signal has been corrupted, probably because the remote is too far away from the rece
iver.
    if(results.decode_type != UNKNOWN) {
      // If a button is on, program that LED with the current IR signal and turn the LED on.
      if(digitalRead(recButton1) == HIGH) {
        lastLed1 = millis();
        ir1 = results;
      }
      // If a button is off, check to see if the current IR signal matches the programmed IR signal.
 If so, turn the LED on.
      else {
        if(results.decode_type == ir1.decode_type && results.value == ir1.value && results.bits
== ir1.bits) {
          lastLed1 = millis();
        }
      }
      if(digitalRead(recButton2) == HIGH) {
        lastLed2 = millis();
        ir2 = results;
      }
      else {
        if(results.decode_type == ir2.decode_type && results.value == ir2.value && results.bits
== ir2.bits) {
          lastLed2 = millis();
        }
      }
      if(digitalRead(recButton3) == HIGH) {
        lastLed3 = millis();
        ir3 = results;
      }
      else {
        if(results.decode_type == ir3.decode_type && results.value == ir3.value && results.bits
== ir3.bits) {
          lastLed3 = millis();
        }
      }
    }
  }

  // Check to see if lastLed has been set in the last 150 milliseconds. This will occur if a button w
as pressed while receiving an IR signal, or if an IR signal matched a programmed signal.
  if((millis() - lastLed1) <= 150) {
    digitalWrite(ledPin1, HIGH);
  }
  else {
    digitalWrite(ledPin1, LOW);
  }
  if((millis() - lastLed2) <= 150) {
    digitalWrite(ledPin2, HIGH);
  }
  else {
    digitalWrite(ledPin2, LOW);
  }
  if((millis() - lastLed3) <= 150) {
    digitalWrite(ledPin3, HIGH);
  }
  else {
    digitalWrite(ledPin3, LOW);
  }
}
```