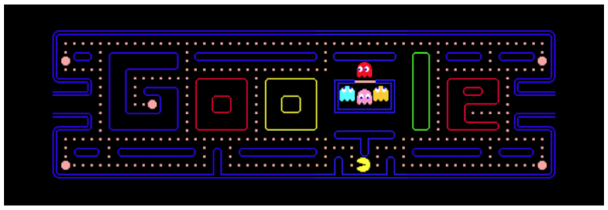
# PacMan Go! | 5



Electrical and Computer Engineering 5

## PacMan Go!

Developed by Kelvin Tran • Nakhul Kisan • Sean Staskus

## Overview

Welcome to Lab 5! This assignment will be used to summarize many of the software tools and hardware designs learned in ECE 5 and as a gentle introduction to OnShape. The tools and techniques that we've learned in class are not only useful in academic and industrial settings, but also to a variety of other fields including visual arts, and video game design. By combining the use of these engineering tools along with components from circuit design, we can effectively set up a game in the MATLAB GUI controlled by an IR remote and sensor.

## What You Will Need

### Materials:

- RGB LEDs
- Red LED, Green LED, Yellow LED
- Jumper wires
- Piezo Buzzer
- Resistors
- Breadboard
- Arduino MEGA
- IR Remote and Sensor

### Machinery:

- 3D Printer

### Software:

- MATLAB
- Arduino IDE
- OnShape
- EAGLE CAD

# Challenge #1: Traffic Light Circuit

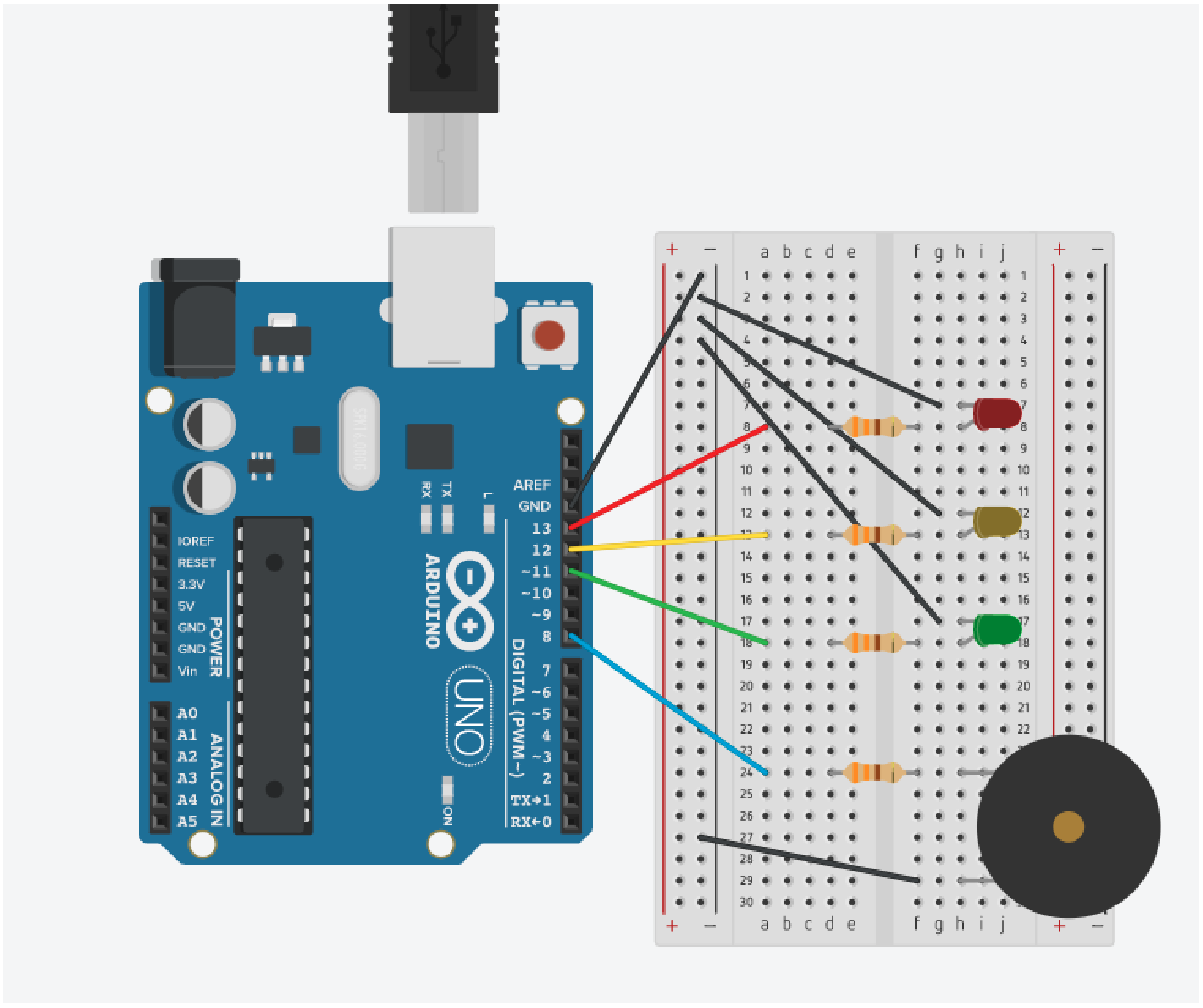
## Objective

The objective is to create a circuit with 3 LEDs in parallel, and line the colors up so that they resemble a traffic light (red on top, green on the bottom, yellow in between; see diagram). Build a piezo buzzer in parallel so that it beeps when the light turns red.

## Components

- 1 Arduino MEGA
- 1 breadboard
- 1 piezo buzzer
- 9 wires (at least 1 black for GND)
- 4 resistors (330 ohms)
- 3 LED lights (1 Red, 1 Yellow, 1 Green)

## Circuit Diagram



**Note:** Refer to the CH1 in Lab 0.  This circuit has 3 LEDs in parallel and is similar to the circuit in that challenge.

## Task

- The circuit should start at the green light and will be on for a random amount of time between 5 and 15 seconds (see code)
- When the light is green, you are allowed to complete the game and play
- The yellow light will flash for 2 seconds and serve as a warning that the red light is coming
- In case you are focused on the game, the piezo buzzer will beep when the light has turned red
- The red light will last for a random amount of time between 3 and 8 seconds
- When red, the game pauses.  You cannot move, but you can look at the screen to plan what your next moves could be

## Instructions

1. Arduino shown in **Fig 2.1**.
2. Hook up the resistors and 3 LED's on a breadboard shown in **Fig 2.2 below**. Space them evenly and use 330 Ω (orange-orange-black-gold).

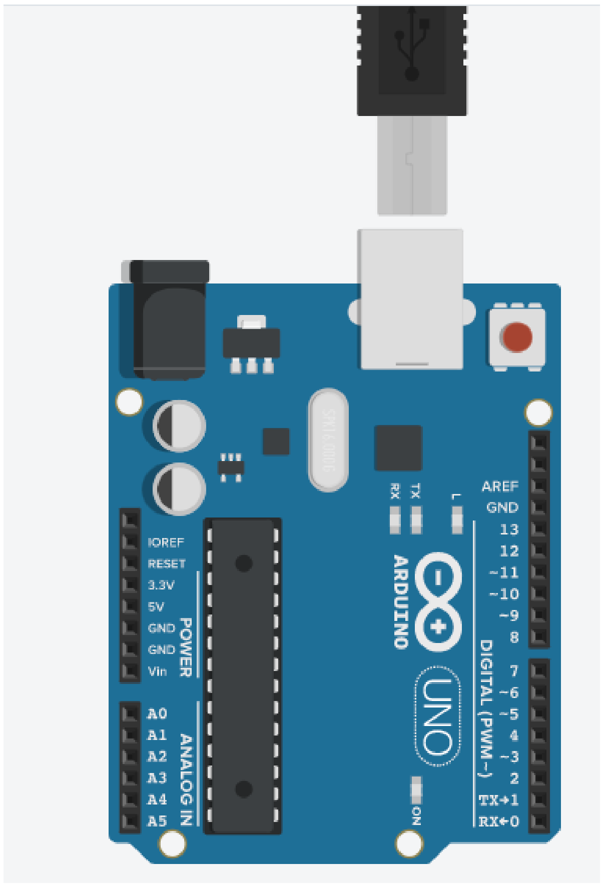


**Figure 2.1**

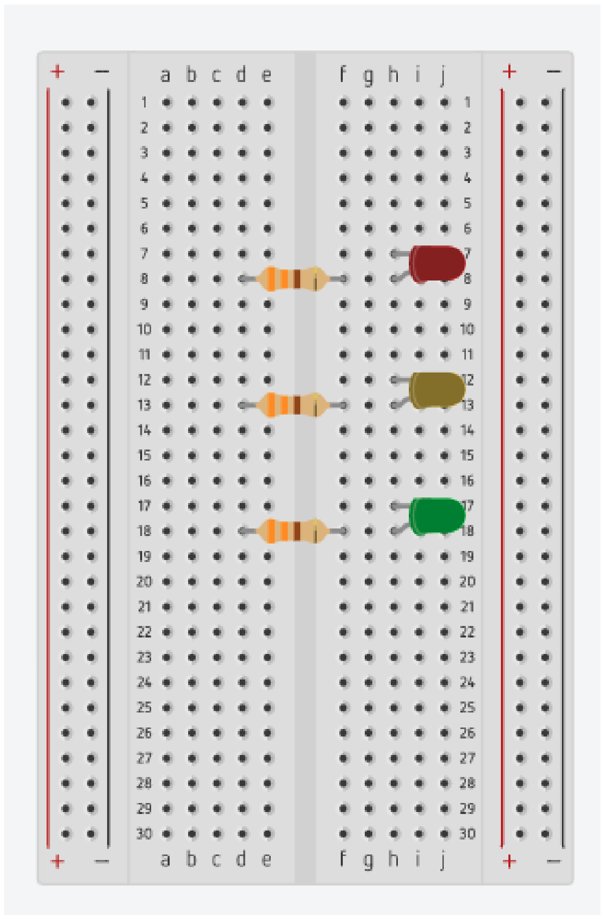


**Figure 2.2**

3. Connect all the cathodes of the LEDs to the minus strip on the breadboard. Connect a wire from the minus strip to the GND on the Arduino. Connect the resistors (not the resistor side closest to the LED's) to 3 separate pins on the digital section of the Arduino board. See **Fig 2.3.**

3. Add the piezo buzzer circuit in parallel with the LED circuit, as seen in **Fig 2.4.** The positive terminal of the piezo should connect to the resistor.

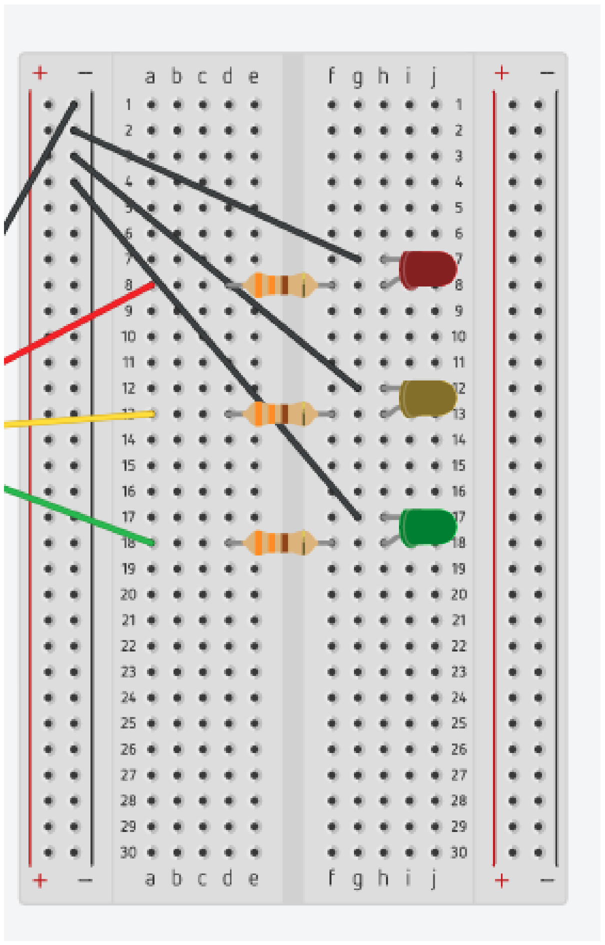


**Figure 2.3**

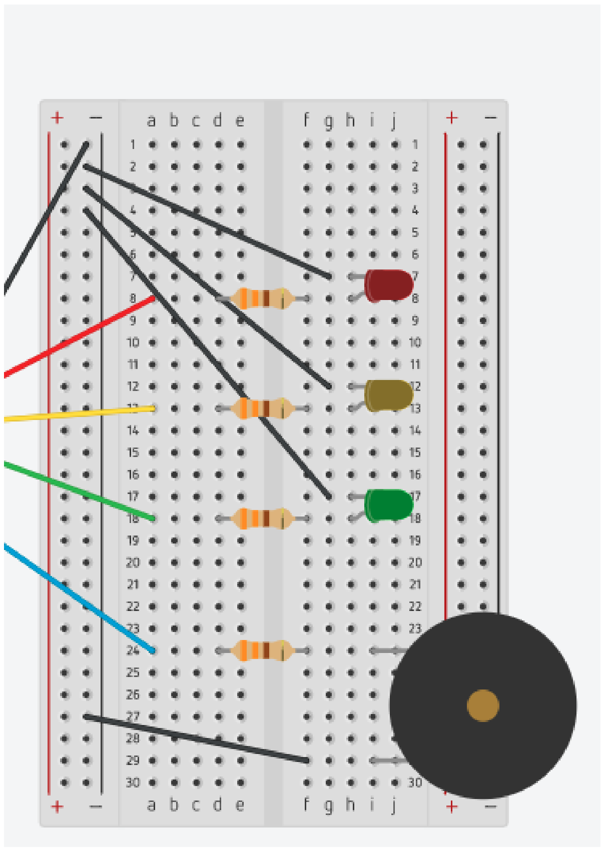


**Figure 2.4**

## {≡} Code

```
// assign the pin value for each LED you are connecting to
int ledR =   ; //red
int ledY =   ; //yellow
int ledG =   ; //green

//set up piezo pin for red light
const int buzzer =   ;

//set up random delay for red and green lights
double randR;
double randG;

void setup() {
 // initialize the digital pin as output.
 // assign each LED pin and piezo as an output
 Serial.begin(    );
 randomSeed(analogRead(0)); //sequence starts at a different place every time

 pinMode(    , OUTPUT);
 pinMode(    , OUTPUT);
 pinMode(    , OUTPUT);

 pinMode(      , OUTPUT);
}
```

```
void loop() {
//start on green to go

 digitalWrite(    , HIGH);// turn the green LED on (HIGH is the voltage level)

  randG = random(5000,150000); //chooses a random number between 5 and 15 seconds for the delay before
changing to yellow

 Serial.print("Green light length: ");
 Serial.print(     / 1000);
 Serial.println(" seconds"); //view Serial to see the precise delay
 delay(    ); //random delay

 digitalWrite(    , LOW); //
 delay(100); // 0.1 sec delay from green to yellow

//after green light, we get a yellow light which is a fixed 2 seconds
 digitalWrite(    , HIGH);
 delay(2000);
 digitalWrite(    , LOW);
 delay(100);

//a red light always follows a yellow light, and the length of the red light is also random
 digitalWrite(    , HIGH);

//sound buzzer
tone(      , 1000);
delay(1000);
noTone(      );

 randR = random(3000,8000); //random delay between 3 seconds and 8 seconds
 Serial.print("Red light length: ");
 Serial.print(     / 1000);
 Serial.println(" seconds"); //view Serial to see the precise delay
 delay(    );
 digitalWrite(    , LOW);
 delay(100);

 //repeats the loop by going from a red light to a green light,
 //which is what traffic lights do

}
```

## Challenge #2: OnShape

**Objective**
The objective of this challenge is to create a shell or encasement device to put your game's hardware into.
**Components**

- OnShape Website or other 3D design software preference

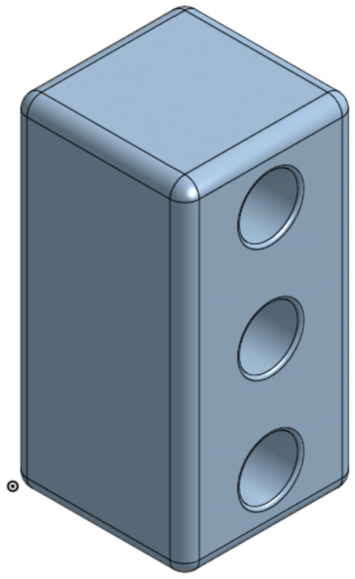
### Task:

- After creating an account on OnShape or downloading your preferred 3D design software, play around with it and ensure you are comfortable with it
- A self-paced guide to learning OnShape can be found here

### Instructions

1. Of course, to create this shell, you have to have your software open, so make sure you do that.
2. From here, take into account any parts you want to put into this encasement, measure them, and consider them in your design
3. That's it. That is all the specifics you are given, so feel free to let your creativity flow. The example below is a bare minimum stoplight shell that is only made to hold the LED lights.

### Example



# Challenge #3: MATLAB Functions

**Objective**

The objective is to create an IR sensor and remote control capable of rotating the LED colors. Also, we wish to create a Matlab function capable of controlling an Arduino's LEDs.

**Components**

- 1 IR sensor
- 1 IR remote
- Matlab
- 2 resistors (330 ohms)
- IR LED
- Jumper Wires
- Circuit from Challenge 1

## Task:

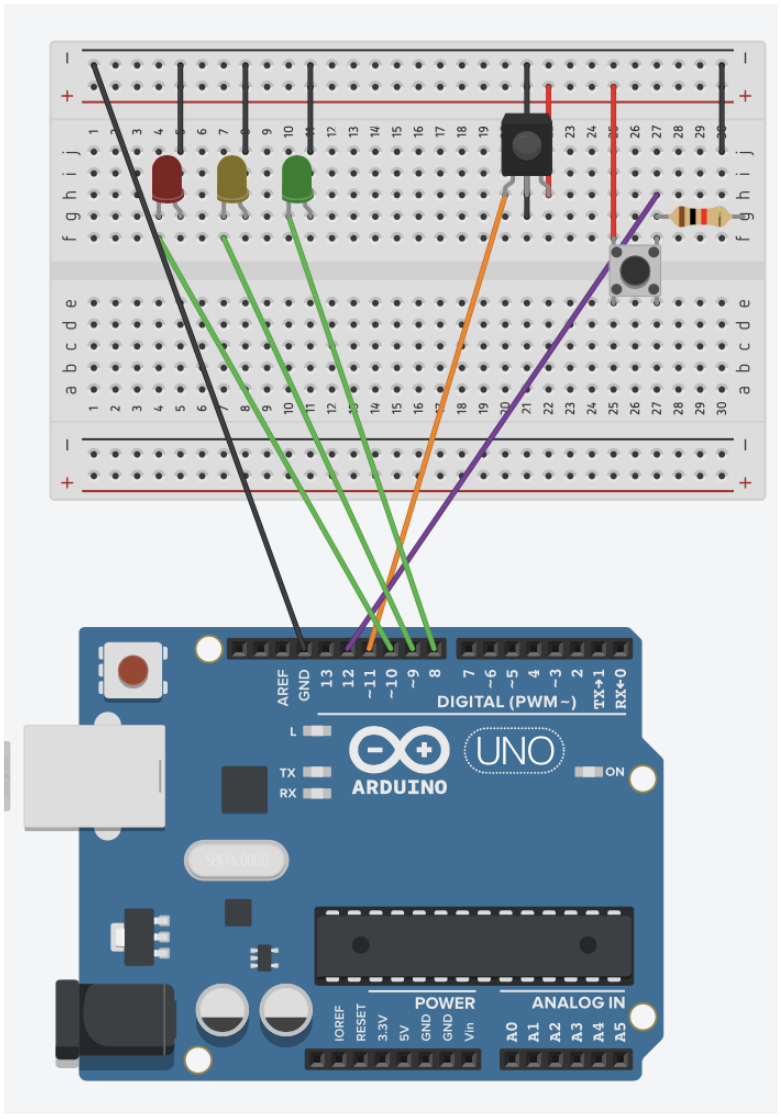
- After downloading the MATLAB Support Package for Arduino Hardware by going to Home > Add-Ons > Get Add-Ons to open the add-on explorer, then searching for "MATLAB Support Package for Arduino Hardware"
- Install the package, click "Setup Now", setup to USB, choose board type, and click "Program"
- Test connection to ensure Arduino is connected, and head to MATLAB
- Do the same for "Sensor Fusion and Tracking Toolbox" found on the Add-Ons menu

## Instructions

Build the following circuit for our IR sensor and remote. Control the LEDs' state using the IR sensor and remote. Then, fill in the MATLAB functions to randomly set off the LEDs.

1. Arduino shown in **Fig 2.1**
2. Hook up the IR sensor as shown below and use 330 Ω (orange-orange-black-gold).

**Figure 2.1**



1. Run through IRrecord example in Arduino IDE (File > Examples > IRremote > IRrecord), as done in lab 1. Record the values picked up by the IR sensor when the 'A', 'B', 'C',, and 'O' buttons are pushed
2. Run through IRrecord example in Arduino IDE (File > Examples > IRremote > IRrecord), as done in lab 1

## {=} Code — IR Sensor Control the LEDs

```
#include <IRremote.h>
#define A 0xF___
#define B 0xF___
#define C 0xF ___
#define ON_OFF_KEY 0xF ___



const int RECV_PIN = ___
const int redPin = ___
const int greenPin = ___
const int yellowPin = ___ ;


int red = 0;  // Initial intensity
int green = 0;
int yellow = 0;
int currentBrightness = 0;  // Initial active color intensity
String targetColor = "RED"; // Initial active color
bool isOn = false;

IRrecv irrecv(RECV_PIN);  // Initialize IR Library
decode_results results;   // Initialize IR Library


void setup() {
  Serial.begin(9600);   // Start Serial
  irrecv.enableIRIn();  // Start the receiver
  // Set the three LED Pins as outputs
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(yellowPin, OUTPUT);
//  pinMode(piezo, OUTPUT);
}



void loop() {
  if (irrecv.decode(&results)) {
    // Serial.println(results.value, HEX);  // Prints HEX code IR receiver receives
    irrecv.resume(); // Receive the next value
    // Power switch
    if (results.value == ON_OFF_KEY) {
      if (isOn == true) {
        Serial.println("Turning Off"); powerOff();
      }
      else if (isOn == false) {
        Serial.println("Turning On"); powerOn();
        showCurrentActiveColor();
      }
    }

    // Color operations
    if (isOn == true) {
      if (results.value == A) {
        turnRed();
      }

      else if (results.value == B) {
        turnYellow();
      }

      else if (results.value == C) {
        turnGreen();
      }
    }
  }
  delay(100);
}

void turnYellow() {
  digitalWrite(yellowPin, HIGH);
  delay(1000);
  digitalWrite(yellowPin, LOW);
  delay(500);
}

void turnGreen() {
  digitalWrite(greenPin, HIGH);
  delay(1000);
  digitalWrite(greenPin, LOW);
  delay(500);
}

void turnRed() {
  digitalWrite(redPin, HIGH);
  delay(1000);
  digitalWrite(redPin, LOW);
  delay(500);
}


// Print current color setting on serial monitor
void showCurrentActiveColor() {
  Serial.print("Now controlling color ");
  Serial.println(targetColor);
}
// Turn Off LED
void powerOff() {
  analogWrite(redPin, 0);
  analogWrite(greenPin, 0);
  analogWrite(yellowPin, 0);
  isOn = false;
}

// Turn on LED
void powerOn() {
  updateColor();
  isOn = true;
```

```
}
//
// Update LED color to current setting
void updateColor() {
  if (targetColor == "RED")
    red = currentBrightness;
  else if (targetColor == "GREEN")
    green = currentBrightness;
  else if (targetColor == "yellow")
    yellow = currentBrightness;
  setColor(red, green, yellow);
  showColorOnSerialMonitor();
}
// Display current color in Serial Monitor
void showColorOnSerialMonitor() {
  Serial.print("Red ");
  Serial.print(red / 255.0);
  Serial.print("\tGreen ");
  Serial.print(green / 255.0);
  Serial.print("\tyellow ");
  Serial.println(yellow / 255.0);
}
// Output desired voltages to LED
void setColor(int red, int green, int yellow) {
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(yellowPin, yellow);
}
```

1. Open MATLAB and run the below code in a new script
2. Be sure to change arduino type from `'uno'` to `'mega2560'`

## {≡} Code — MATLAB Random LED

```
clear all; close all; fclose all; delete(instrfind); clc; % Start Fresh

% % declare arduino object
ARDUINO_SERIAL_PORT = 'THIS'; % Change this (Win: 'COM?', MAC: '/dev/cu.usbmodem?')
a = arduino(ARDUINO_SERIAL_PORT, 'BOARD'); % Change this to board type

for i = 0:100
    writeDigitalPin(a,'__',1);
    writeDigitalPin(a,'__',0);

    writeDigitalPin(a,'__',1);
    writeDigitalPin(a,'__',0);

    writeDigitalPin(a,'__',1);
    writeDigitalPin(a,'__',0);

    playTone(a,'__',2400,10);
```

```
end
```

Using the arduino commands, the layout from the above code, and your knowledge of MATLAB function declarations, randomize the length that the Green and Red LEDs go off

## Challenge #4: PacMan GUI

**Objective**

The objective of this challenge is to setup a MATLAB GUI to display our PacMan game

**Components**

- Matlab Online
- Matlab GUI IDE

### Task:

- Download Pacman package from the MATLAB and run through the game
- Load game data into the game window

### Instructions

1. Download the Pacman package by going to Home > Add-Ons > Get Add-Ons to open the add-on explorer, then searching for “Pacman”. Click “Add to Matlab”
2. You should now have a series of matlab functions downloaded into your folder
3. Read through the documentation to understand what each mat file does
4. CLick ‘LoadData’ in the game home-screen and load gameData.m
5. Click ‘New Game’ and run through an iteration of the game

# Challenge #5: PacMan Controls

**Objective**

The objective of this challenge is to setup the game of PacMan GO!

**Components**

- Matlab Online
- Matlab GUI IDE
- Circuit from challenge 1
- Pacman Game from challenge 4

## Task:

- Write MATLAB callback functions to control the LEDs rotation, set off piezo buzzer, keep track of valid and invalid user moves, and run through the game of Pacman.

## Instructions

Note: Use function GameLoop to control when valid input is allowed for the Pacman game

1. In Pacman.m insert the following code in the %% General configurations section of the function pacman. This will connect the game to our arduino board

```
% % declare arduino object
ARDUINO_SERIAL_PORT = '/dev/cu.usbmodem146101'; % Change this (Win: 'COM?', MAC: '/dev/cu.usbmodem?')
a = arduino(ARDUINO_SERIAL_PORT, 'uno'); % Change this to the board being used
redPin = '__';
greenPin =  '__';
yellowPin = '__';
piezo = '__';
redLight = 0;
yellowLight = 0;
greenLight = 1;

writeDigitalPin(a, greenPin,1);
```

1. Copy and paste and the below code into Pacman.m beneath the function GameLoop to control

when the Red, yellow, and green LEDs are lit, and, subsequently, when the game is paused and when the game is on.

## {=} Code — CallBack Functions

```
``function RandGame
        stop(myTimer);
        isPause = 1;
        randRGB = randi(10);
        if(randRGB ==5)
            RedChange;
            start(myTimer);
            isPause = 0;
        elseif(mod(randRGB,3) == 0 | randRGB == 7)
            YellowChange;
            GameLoop;
        else
            GreenChange;
            GameLoop;
        end
    end


    function GreenChange
        % turn off all lights
        writeDigitalPin(a, greenPin,0);
        greenLight = 0;
        pause(0.1);

        writeDigitalPin(a, redPin,0);
        redLight = 0;
        pause(0.1);

        writeDigitalPin(a,yellowPin,0);
        yellowLight =0;
        pause(0.1);



        writeDigitalPin(a, greenPin,1);% turn the green LED on (HIGH is the voltage level)
        greenLight = 1;
        randG = 5+rand(1)*(15-5); %chooses a random number between 5 and 15 seconds for the delay before
changing to yellow
        start(myTimer);
        isPause = 0;
    end


    function RedChange
         % turn off all lights
        writeDigitalPin(a, greenPin,0);
        greenLight = 0;
        pause(0.1);

        writeDigitalPin(a, redPin,0);
        redLight = 0;
        pause(0.1);

        writeDigitalPin(a,yellowPin,0);
        yellowLight =0;
        pause(0.1);


        writeDigitalPin(a, redPin,1);
        redLight = 1;

        %sound buzzer
        playTone(a, piezo,1200,1);

        randR = 1+rand(1)*(4-1); %random delay between 3 seconds and 8 seconds
        pause(randR);

    end



    function YellowChange
        % turn off all lights
        writeDigitalPin(a, greenPin,0);
        greenLight = 0;
        pause(0.1);

        writeDigitalPin(a, redPin,0);
        redLight = 0;
        pause(0.1);

        writeDigitalPin(a,yellowPin,0);
        yellowLight =0;
        pause(0.1);


        % we get a yellow light which is a fixed 2 seconds
        writeDigitalPin(a, yellowPin,1);
        start(myTimer);
        isPause = 0;
        yellowLight =1;
    end
```

1. Finally, we need to change how the game's timer runs to include our LEDs.

```
% CHANGE THIS LINE:
myTimer = timer('TimerFcn',@(s,e)RandGame,'Period',game.speed,'ExecutionMode','fixedRate');


% TO BECOME THIS LINE INSTEAD
myTimer = timer('TimerFcn',@(s,e)RandGame,'Period',game.speed,'ExecutionMode','fixedRate');
```

## Reflection

### Challenges

Some challenges we faced during this project were getting the timer and lights to agree with Pac-Man and figuring out how to implement 3D modeling into our lab.

### Experience

We learned a lot about hardware and software agreement during this, as well as 3D design through OnShape and EagleCAD. We also experienced building teams on short notice, similar to what may happen in our jobs later on.