# Arduino Report

Rhys J - 40510730

## Requirements

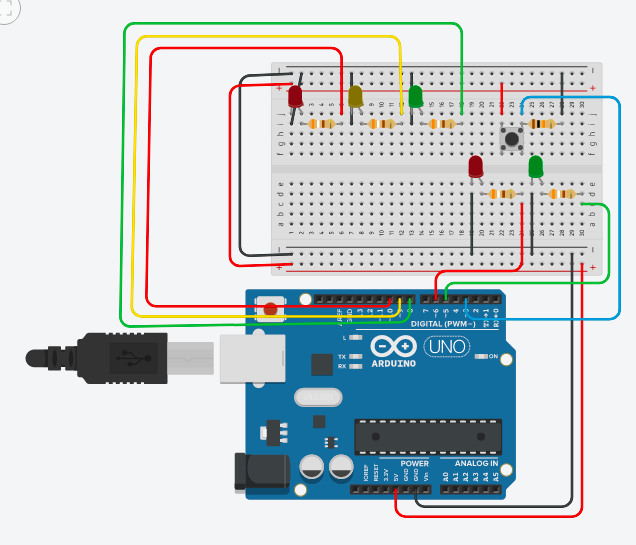
Project Description

I have been tasked to create a prototype pelican crossing. I must accomplish this by using an Arduino, a breadboard, red, amber and green LEDs, a button, and wires. A red, amber and green LED will represent the lights used for the traffic lights that control the flow of vehicles. A separate red and green LED will represent the lights allowing pedestrians to cross the road. The button will be used by the pedestrians wanting to cross. Pressing the button will start a sequence that will stop traffic and allow pedestrians to cross.

Pelican Crossing Sequence

1. By default green light is shown to vehicles and red light is shown to pedestrians.
2. When button is pressed the following sequence will occur; if the button is pressed again it will not be registered, until the amber light is flashing.
3. Amber light shown to vehicles for 3 seconds.
4. Red light shown to vehicles and green light shown to pedestrians for 10 seconds.
5. Amber light flashes for vehicles and green light flashes for pedestrians half a second on and half a second off, both for 5 seconds.
6. The lights are then set to the default where green light is shown to vehicles and red light is shown to pedestrians.
7. Traffic will not be stopped again for 1 minute.

## Design



## Code

// Arduino Pelican Crossing

// Author: Rhys J 40510730

// v1.0

// constants for the pin number connected to the component

const int redLED = 10;

const int yellowLED = 9;

const int greenLED = 8;

const int walkRedLED = 6;

const int walkGreenLED = 5;

const int button = 3;

// constants for the stages of the sequence

const int notRunning = 0; // traffic light sequence is not running

const int startSequence = 1; // traffic light sequence is running

const int flashing = 2; // stage of sequence where the yellow and walk green LED are flashing

int time = 0; // timer variable to check 1 minute has passed since traffic was stopped

int sequenceRunning = notRunning; // variable stores the current stage of the sequence

bool buttonPressed = false; // variable if button has been pressed

// this function is run when the button has been pressed, due to the interrupt

void ButtonPressedInterrupt()

{

// button press is only registered when the sequence is not running or is flashing

if(sequenceRunning != startSequence)

{

buttonPressed = true; // button press is registered

}

}

// this function is run once when program is started

void setup()

{

// setup all the connected pins

pinMode(redLED, OUTPUT);

pinMode(yellowLED, OUTPUT);

pinMode(greenLED, OUTPUT);

pinMode(walkRedLED, OUTPUT);

pinMode(walkGreenLED, OUTPUT);

pinMode(button, INPUT);

// when button pin goes from high to low the specified function is run

attachInterrupt(digitalPinToInterrupt(button), ButtonPressedInterrupt, FALLING);

// setup the default lights that are on

digitalWrite(greenLED, HIGH);

digitalWrite(walkRedLED, HIGH);

}

// function is repeated indefinitely while the program is running

void loop()

{

// decrements the timer by 1 every second

delay(1000);

time--;

// sequence will run if timer is 0 or less and when button press has been registered

if (time <= 0 && buttonPressed)

{

buttonPressed = false; // button press is no longer pressed

sequenceRunning = startSequence;

digitalWrite(greenLED, LOW);

digitalWrite(yellowLED, HIGH);

delay(3000);

digitalWrite(yellowLED, LOW);

digitalWrite(redLED, HIGH);

digitalWrite(walkRedLED, LOW);

digitalWrite(walkGreenLED, HIGH);

delay(10000);

digitalWrite(redLED, LOW);

sequenceRunning = flashing; // button presses from this point will be registered

// loop will repeat 5 times making the lights flash

for(int i=0; i<5; i++)

{

digitalWrite(yellowLED, LOW);

digitalWrite(walkGreenLED, LOW);

delay(500);

digitalWrite(yellowLED, HIGH);

digitalWrite(walkGreenLED, HIGH);

delay(500);

}

digitalWrite(greenLED, HIGH);

digitalWrite(walkRedLED, HIGH);

digitalWrite(yellowLED, LOW);

digitalWrite(walkGreenLED, LOW);

time = 60; // traffic will not be stopped for 60 seconds

sequenceRunning = notRunning;

}

}