Automated Live Trap

Project Report

May 5, 2017

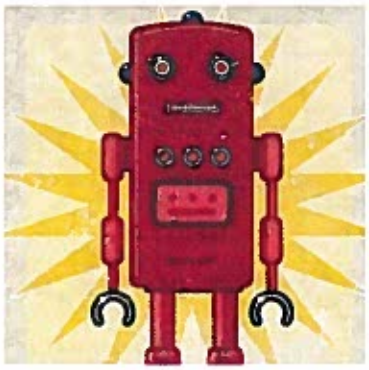
Group 53

Bryan Goings

Michael Krause

Jason LeJeune

Samuel Thornberry



+

Table of Contents

Design Summary

Live traps are commonly used in a wide range of applications, primarily for humane removal of pests or for field research purposes. Currently, live traps on the market function in a strictly mechanical manner, requiring significant user input and interaction. To address this common issue, a live trap was constructed that retains all of the important aspects of currently available models, while implementing a range of mechatronic systems to reduce the overall required user interaction.

A master power switch is used to turn the system on and off. When "on" the system will boot, with the trap in safe mode, meaning the motor will not keep the gate open or closed. A second switch will be used to arm or disarm the trap, this switch will allow the motor to be controlled by the Arduino UNO. When triggered, the gate will close and both audio and visual signals, a red LED and a tone, will indicate that the gate is closed. The reset button raises the gate and triggers a second visual cue, a green LED.

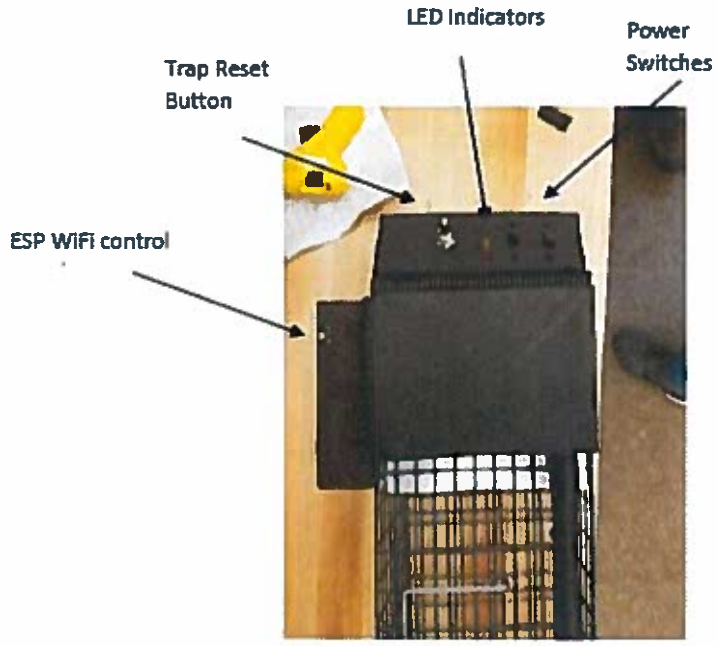


Figure 1: The control panel of the live trap, showing all control switches and the LED's

This trap utilizes a small NEMA 17 stepper motor to operate a gate that closely resembles that found on standard live traps. Using a system of gears affixed to both the motor and the gate, the trap will automatically open and close upon receipt of a specific signal from an Arduino microcontroller. To set the trap, the Arduino is interfaced with both a manual user input (button) and WiFi module. By pressing either the button affixed to the trap, or indicating it remotely via WiFi, the Arduino will signal the stepper motor to open the gate to arm the trap.

Figure 2: The NEMA 17 motor and gears at the front of the trap are shielded by a 3D printed cover.

A PIR sensor will be located towards the rear of the trap that will be directly connected to the Arduino. Upon detecting movement of a curious animal, a signal is sent to the Arduino. After receiving this signal, the Arduino triggers the stepper motor to close the gate of the trap and triggers audio and visual cues. At any time the user can use the remote interface to either stream video of the trap or take single still photos. Additionally, if an undesired animal is caught the user can use the remote interface to raise the gate and set the animal free, and then reset the trap.

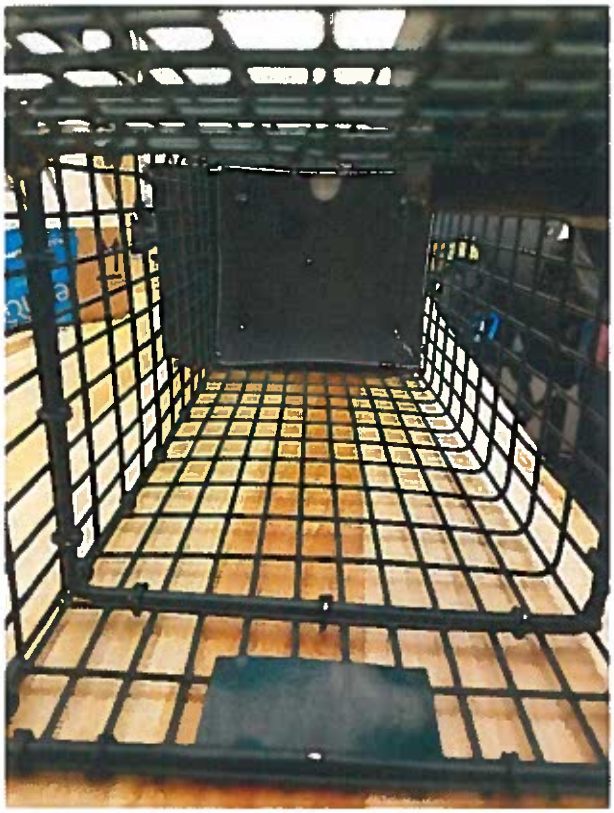


Figure 3: A view of the PIR motion sensor used to detect motion in the trap.

System Details

In this project, a live trap was mechanized for easier use. Figure 4 below shows the basic layout of the different components. The PIC controls the LED's and interfaces with the Arduino. The ESP receives input from the camera, and interfaces with the Arduino. The Arduino receives power from the switch, receives input from the gate control button, and controls the speaker and stepper motor. The Stepper motor uses a gear train to open the gate.

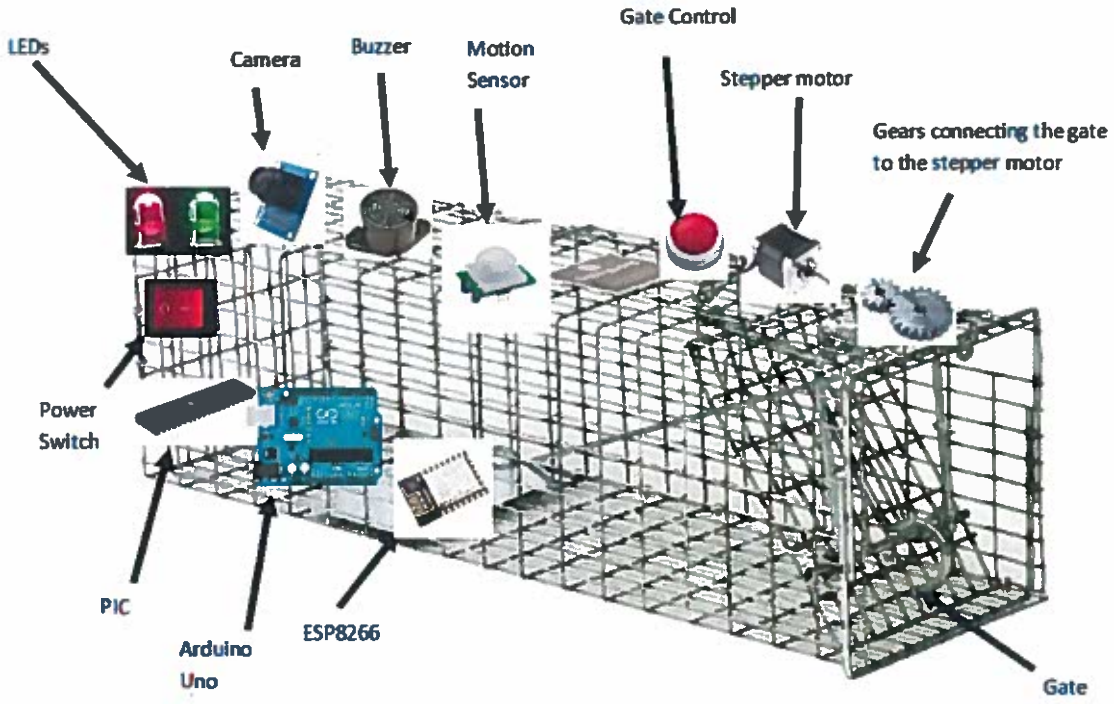


Figure 4: Diagram showing components of live trap

The gate subsystem of the device is a core feature that directly controls both the open and close behavior according to certain inputs. A simple gear train is connected directly to the gate and is utilized to transmit torque from a NEMA 17 stepper motor. This system is housed within a 3D printed housing, as seen in the left hand side of Figure 2. This gear train system is not only responsible for opening and closing the gate, but also maintaining said position to prevent the

escape of the trapped animal or accidental closure. To control this rotational motion, the stepper motor is connected to the A4988 driver module, shown below in Figure 5.

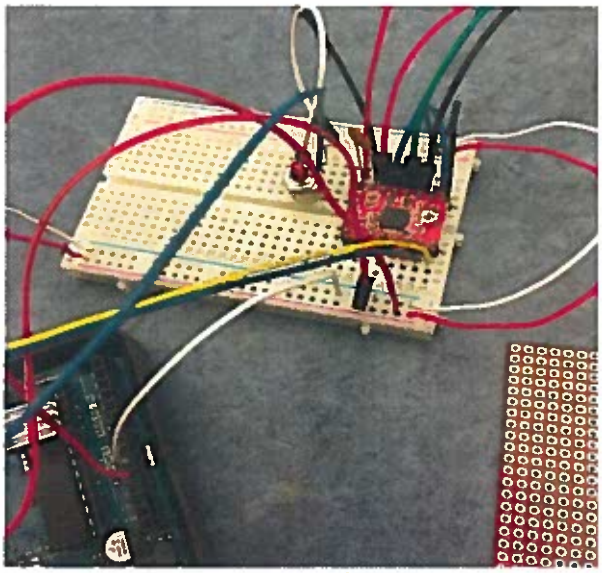


Figure 5: A4988 Driver Module connected to stepper motor and Arduino Uno

The A4988 driver module contains logic that allows 2 input signals from an Arduino Uno to be converted into directional rotation of the stepper motor. The Arduino is programmed to send signals to these two driver inputs when a certain manual input (WiFi or direct manual input) or signal from the PIR sensor is received. One signal controls the direction of the stepper motor, while the other signal dictates how many steps, or pulses, the motor must execute.

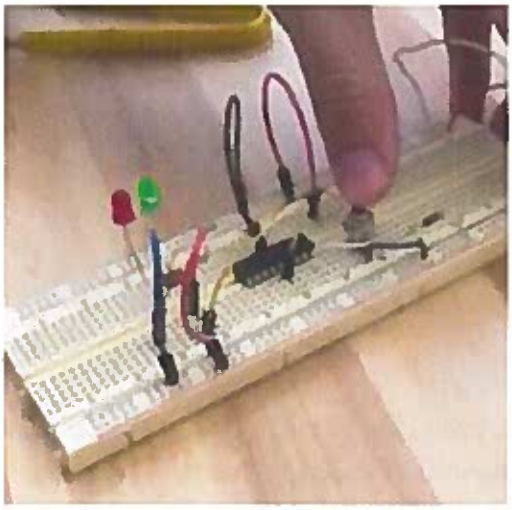


Figure 6: PIC16F88 circuit used to control LED output

Figure 6 shows the pic circuit controlling the two LEDs. The red LED is connected to port B1 and the green LED is connected to port B0 of the pic. Initially the green LED is lit up while the red LED is off. While the button in figure 6 is pressed down, the the green LED turns off and the red LED begins to flash. The button is replaced in the finished product as a signal received by the pic from the arduino once the trap gate closes.

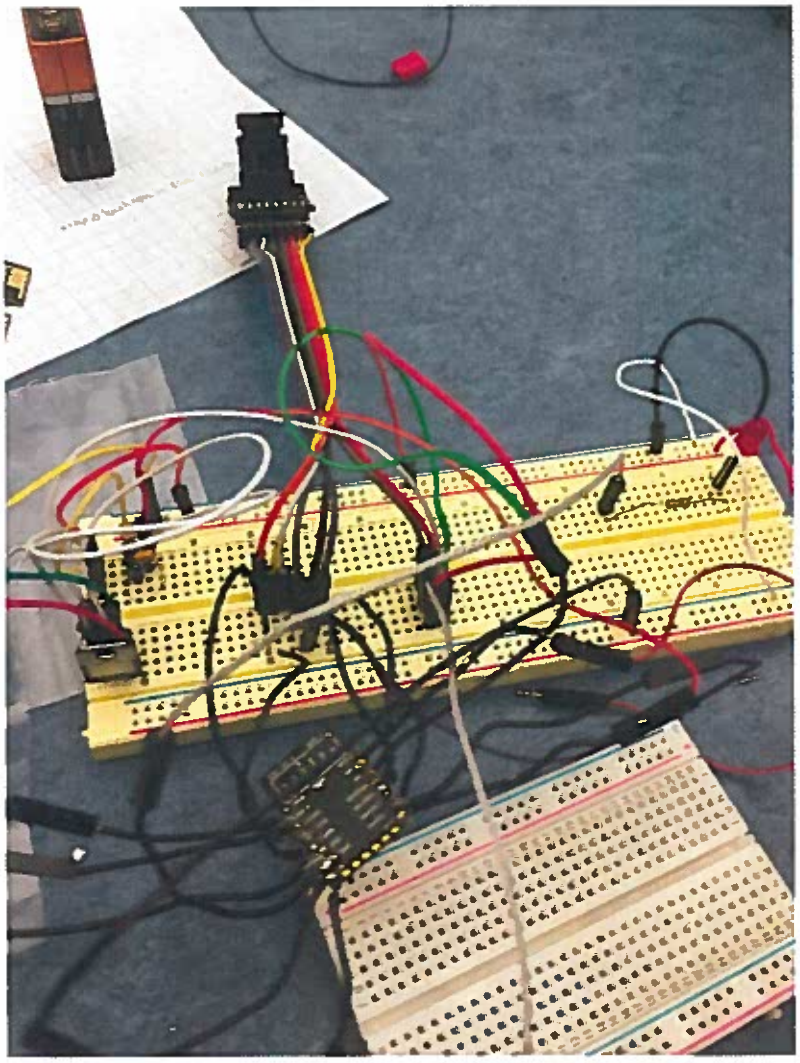


Figure 7: ESP8266 and ArduCAM

As seen in Figure 7 above, the ArduCAM is connected to the ESP8266. The ESP8266 can then send a livestream of the cage via wifi by entering the local IP address of the ESP8266 into a web browser with the handle "stream". If the IP address is entered with the handle "button", the ESP8266 will send a signal to the Arduino, which then opens or closes the gate. The ESP8266 receives power from a 3.3V voltage divider. This voltage divider has an input of 5V from the Arduino.

A functional diagram for the project can be seen in figure 8 below. This diagram shows how the two microcontrollers are connected to the rest of the system.

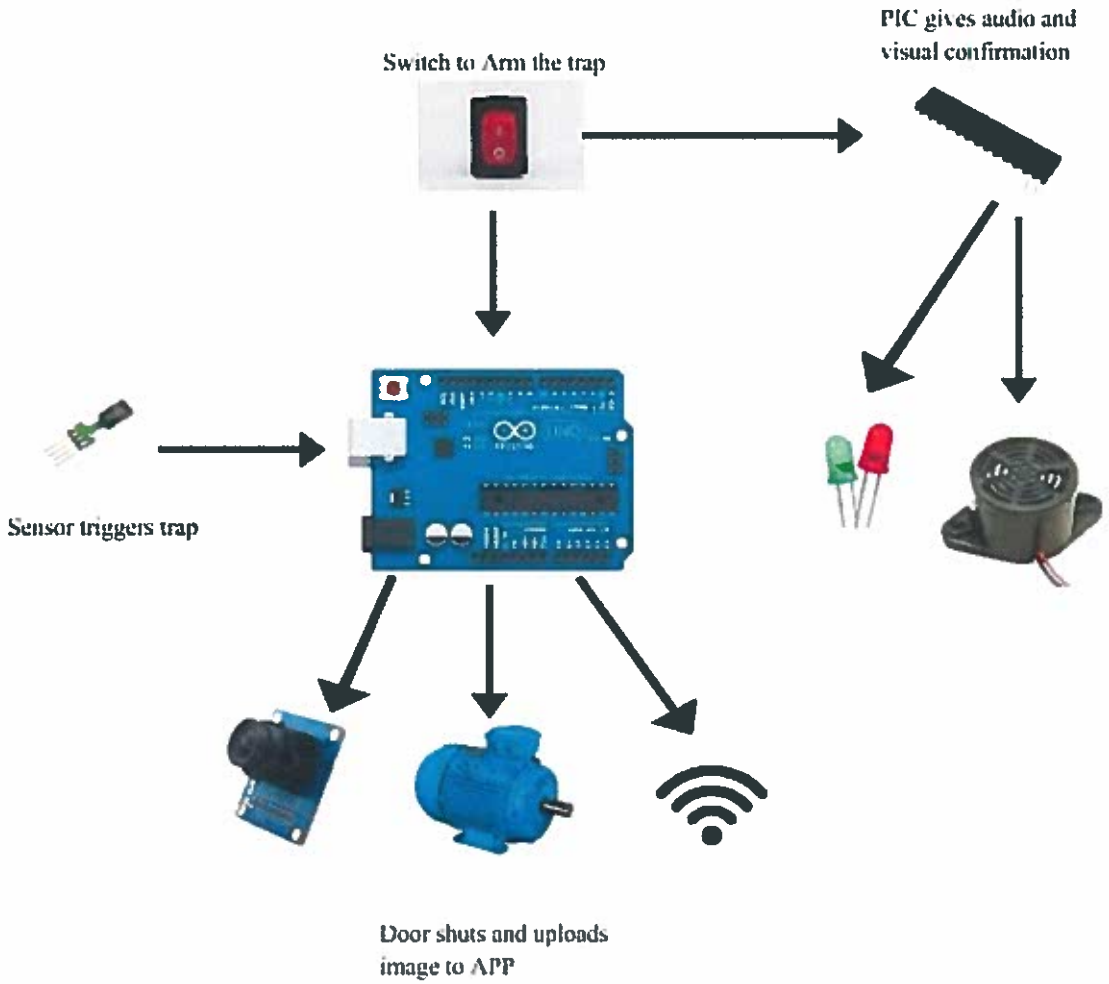


Figure 8: Functional diagram for the automated live animal trap.

A software flowchart for the project is shown below in figure 9. This flowchart demonstrates the process that the animal trap goes through as certain user and non-user inputs are activated. The flowchart explains the pic and the arduino microcontrollers.

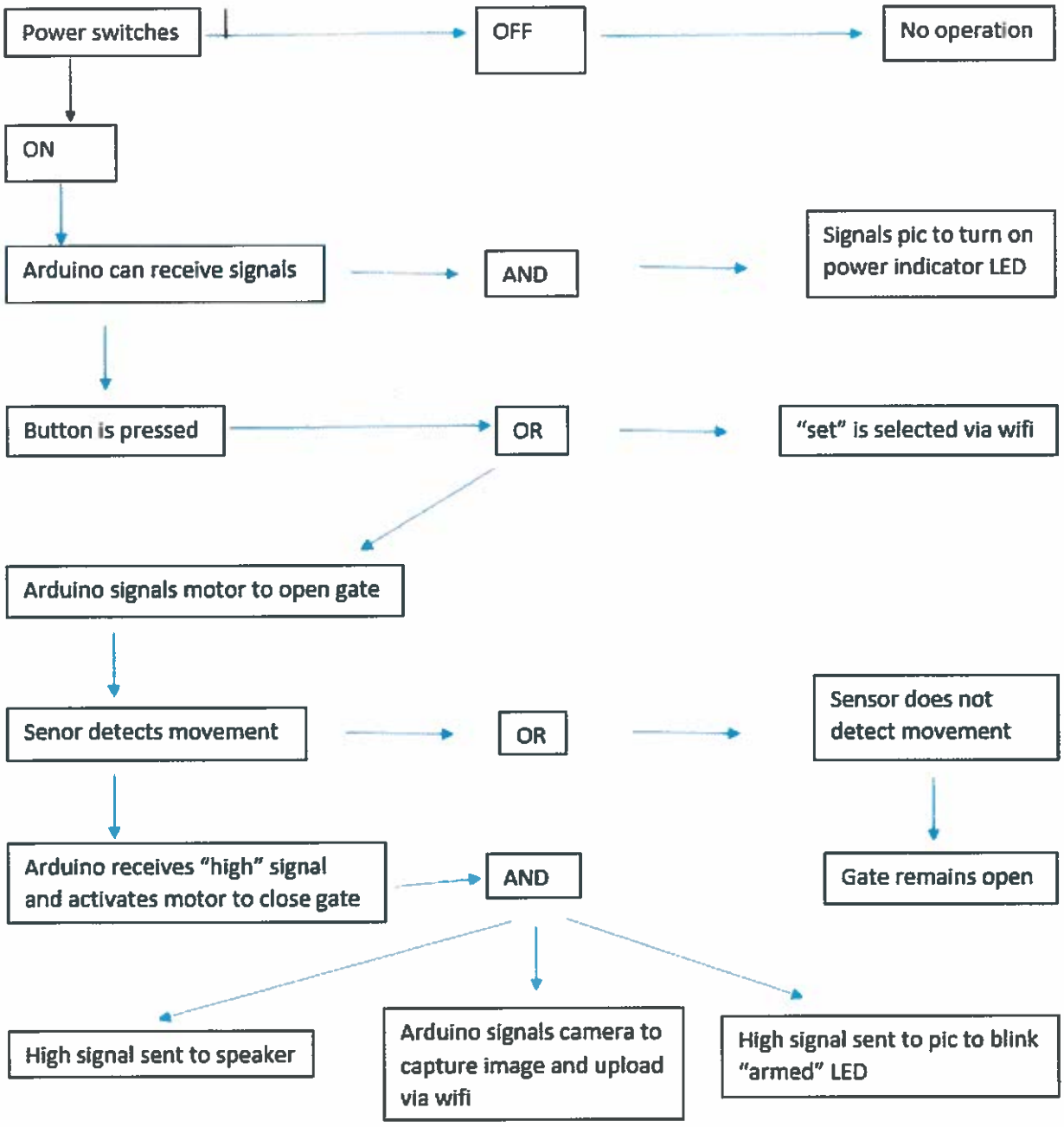


Figure 9: Software flowchart for project

Appendices A and B include wiring diagrams for the pic and the Arduino circuits respectively. The wiring diagrams were drawn using free software from Fritzing. The code used to program

the pic is shown in Appendix D and the arduino code can be found in Appendix E. There is also a bill of materials listed in appendix C.

Design Evaluation

Required functional element category A: Output Display was met in the form of two LEDs and a video/photo stream. One LED with a solid green light denotes that the device has power and is ready to be used. Once the motion sensor detects motion and causes the animal trap gate to shut, the solid green LED turns off and a flashing red LED turns on. This red LED continues flashing until the animal trap gate is opened, and at which time the red turns off and the green turns back on. A Pic16f88 controls the LEDs (code in appendix) and communicates with an Arduino Uno to know when the motor has closed the gate signifying that the green LED should turn off and red LED should turn on. This shows a functionality, repeatability, and decent effort. An LED being controlled by a pic was discussed in the lab, but using multiple LEDs controlled by a pic that also communicated with an Arduino to supply the high or low signal for the LEDs required out of class research and learning. Additionally, an ArduCam Mini was interfaced with an ESP8266 WiFi module to provide a visual output of the interior of the cage streamed to a laptop. To achieve this function, significant research was required to understand how to interface the camera, ESP8266, Arduino, and personal computer seamlessly.

Required functional element category B: Audio Output Device was met in the form of an 8 Hz speaker with software-generated sound effects. The speaker was connected to an Arduino Uno with the melody made from different pitches and note durations. The code can be found in the appendix. The speaker plays the melody every time the animal trap gate is closed. This speaker melody is both functioning and repeatable and required a level of added effort. The hardware was easily connected to the Arduino, but software in the form of the pitches and lengths of notes required some research into which pitches sounded pleasant to the human ear, and what lengths would produce the melody desired.

Required functional element category C: Manual User Input was met through multiple toggle switches and a push button. The first toggle switch provides power to the microcontrollers (Pic16f88 and Arduino Uno). The second toggle switch turns on the motor and allows the animal trap gate to open and close by the stepper motor. The third toggle switch is used to connect the ESP8266 module to the Arduino. When the ESP8266 is connecting to WIFI, pin 2 needs to be disconnected. Once the ESP8266 has connected, the switch can be flipped. The push button allows the user to open the gate once it has been closed. These manual user inputs are both functioning and repeatable. The fact that there are multiple user inputs producing different outputs demonstrates the fact that more of an effort was put into this category than simply creating a switch that turned on the power and stopping there. An additional manual user

input is the ability to open and close the gate from a web browser. The ESP8266 receives a signal via WIFI. It then communicates with the Arduino to control the motor. This manual user input functions, is repeatable, and required substantial independent research as it was not covered in class, the lab, or the textbook.

Required functional element category D: Automatic Sensor was met through the use of a proximity sensor attached in the back of the trap. The proximity sensor is hooked up to the Arduino and once it detects motion, it allows for a signal to be sent to the stepper motor to close the trap gate. This automatic sensor is functioning, repeatable, and required significant research to successfully integrate with the rest of the system. Proximity sensors were not covered in lab and very briefly touched on in the textbook which caused the group to do independent research to learn about this type of automatic sensor.

Required functional element category E: Actuators, Mechanisms and Hardware was met through the use of a stepper motor. The motor and driver are connected to the Arduino. Once the stepper motor receives a signal from the Arduino, it opens the trap gate. This signal comes from either the wifi or push button manual user inputs. Once the proximity sensor detects motion, a signal is sent from the Arduino so that the motor closes the trap gate. The stepper motor is both functional and repeatable. Stepper motors were not covered in the lab, and were covered in one lecture of the semester. Due to the lack of in depth coverage of this type of actuator, significant research was required to achieve successful integration of the motor and driver with the rest of the project. The stepper motor selected was a NEMA 17 bipolar version as bipolar stepper motors offer greater torque due to energizing all coils with alternating polarity such that all coils actively turn the motor. Each step of the motor was determined to be 1.8 degrees from the manufacturer data sheet, so a rough calculation was performed to determine the motor step count to be executed by the code. Originally approximately 80 steps were specified, however after testing the system, the step count was fine tuned to 95 steps. Although 90 degrees of motor rotation is only approximately 50 steps, a greater value of steps were needed due to the gear reduction of the gear train.

Required functional element category F: Logic, Processing, and Control; AND Miscellaneous (functional elements not covered in the categories above) was met through the use of open-loop control. No feedback is given to indicate that the trap gate is opened. The push button or wifi can trigger the gate to open even if it is currently in an opened state. If this occurs, an audible noise can be heard which is the motor trying to open the gate. This essentially does nothing and the system continues to operate as normal. The outputs of the system are not fed back to the inputs.

An Arducam Mini camera and OV5642 camera shield was also connected to the back of the trap. Upon command the camera can either take a photo or stream video of the trap to an IP address, via wifi, that the user can access to view and save the image or video. The open loop logic is functional and repeatable with moderate additional required effort. The camera was not covered in lab, class, or the book and required substantial independent research and effort for successful integration.

Partial Parts List

Table 1: Partial parts list for project

	Item	Quantity	Total Price	Retailer
1	Stepper Motor	1	$13.50	Amazon
2	Motor Driver	1	$5.69	Amazon
3	Camera	1	$39.99	Amazon
4	ESP8266	1	$6.95	Sparkfun
5	PIR Sensor	1	$9.95	Sparkfun
6	Wire Jumpers	1 set of 10	$3.90	Sparkfun
9	Live Trap	1	$31.99	JAX Farm Supply
11	Heat Shrink	1 package	$3.98	Home Depot
12	Split Loom	1 coil	$4.89	Home Depot
15	Speaker	1	$4.75	Mountain State Electronics
16	Toggle Switch	2	$3.24	Radio Shack
18	5 and 3.3V Voltage Regulators	2	$5.47	Mountain State Electronics
19	Arduino	1	$18.82	Amazon
22	PIC 16F88	1	Free	Microchip
23	M3 Fasteners	1 package	$2.19	Home Depot
25	3d Printer Filament	1	$24.99	Amazon
27	Gears	2	$12.98	Amazon
28	#4 Fasteners	2 packages	$2.53	Home Depot

The table above shows parts used in the project other than common components used in lab or mentioned in class such as LEDs, switches, resistors, etc. as well as the price and retailer. The total project expenses came to a grand total of $290.36 which includes the parts listed above, as well as common components. A full bill of materials can be found in the appendix.

Lessons Learned

Problems faced and solved:

1. We needed a way to cover up the wires and other components of our project without increasing weight by much. We made use of CSU's 3D printing lab which allowed for quick lightweight parts to be produced.

2. Group members not knowing what to do once their assigned task was completed. We solved this by making a list of all remaining objectives and having group members move on to the next one once they were finished with their current one.

3. We needed replacement parts for some electrical components very quickly. We did not have time to wait to have them shipped by online vendors so we made use of local shops including Mountain State Electronics, RadioShack, and Sparkfun (in Boulder).

4. All group members were busy with other classes and extracurricular obligations so it was difficult to find a regular time for everyone to meet every week. This was solved by the group sitting down in person and comparing schedules to find a regular time every week where everyone was available to meet in the lab to work.

5. A few of the components we used were not covered in class and no one felt confident on how to create the circuits successfully. To solve this issue we made use of online forums and tutorials. There were relatable if not very similar examples online for many of the components we were using which was very helpful in troubleshooting.

Recommendations for future students:

1. Start on the mechanical parts as quickly as possible. Most of the electrical and software components will be covered later in the semester.

2. Always buy extra parts for the project in case something breaks unexpectedly (especially electrical components).

3. Make expectations of group members very well known and think about creating a contract that all members sign.

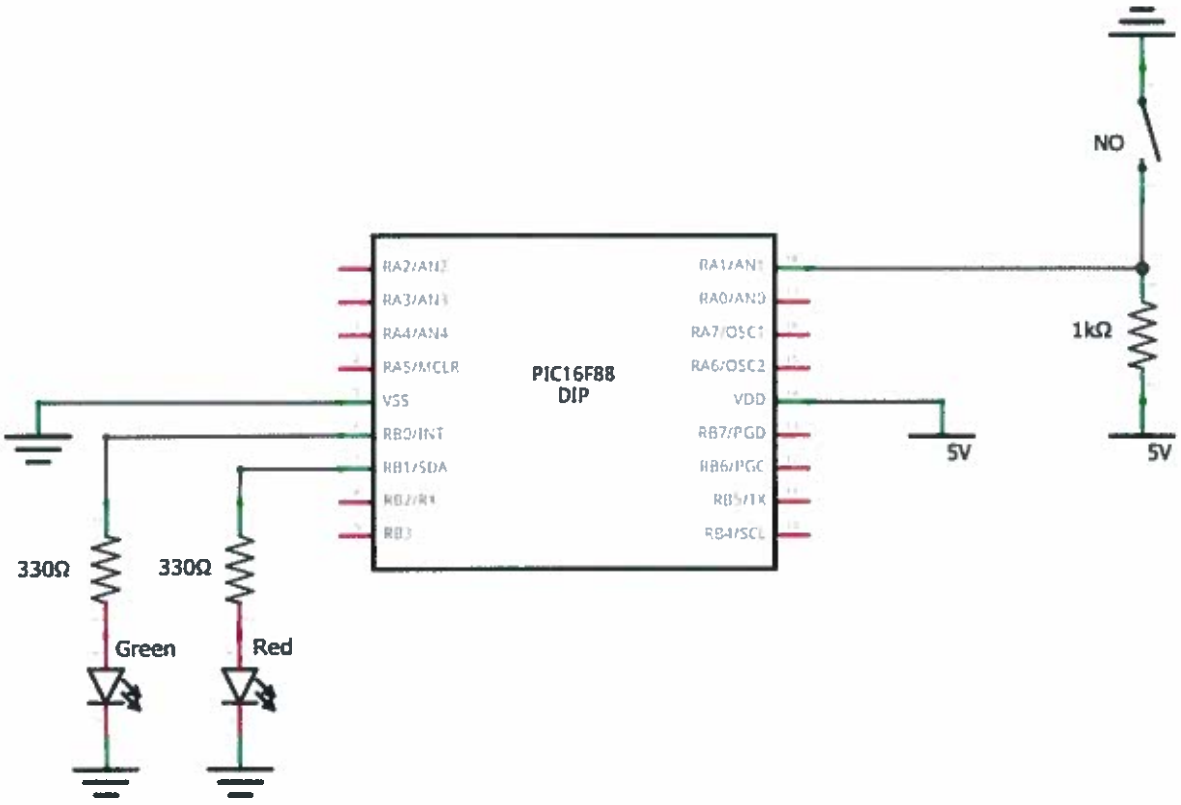
4. Make use of online tutorials on how to create electrical circuits. Most of the circuits used in the project have been built and explained online by a professional.

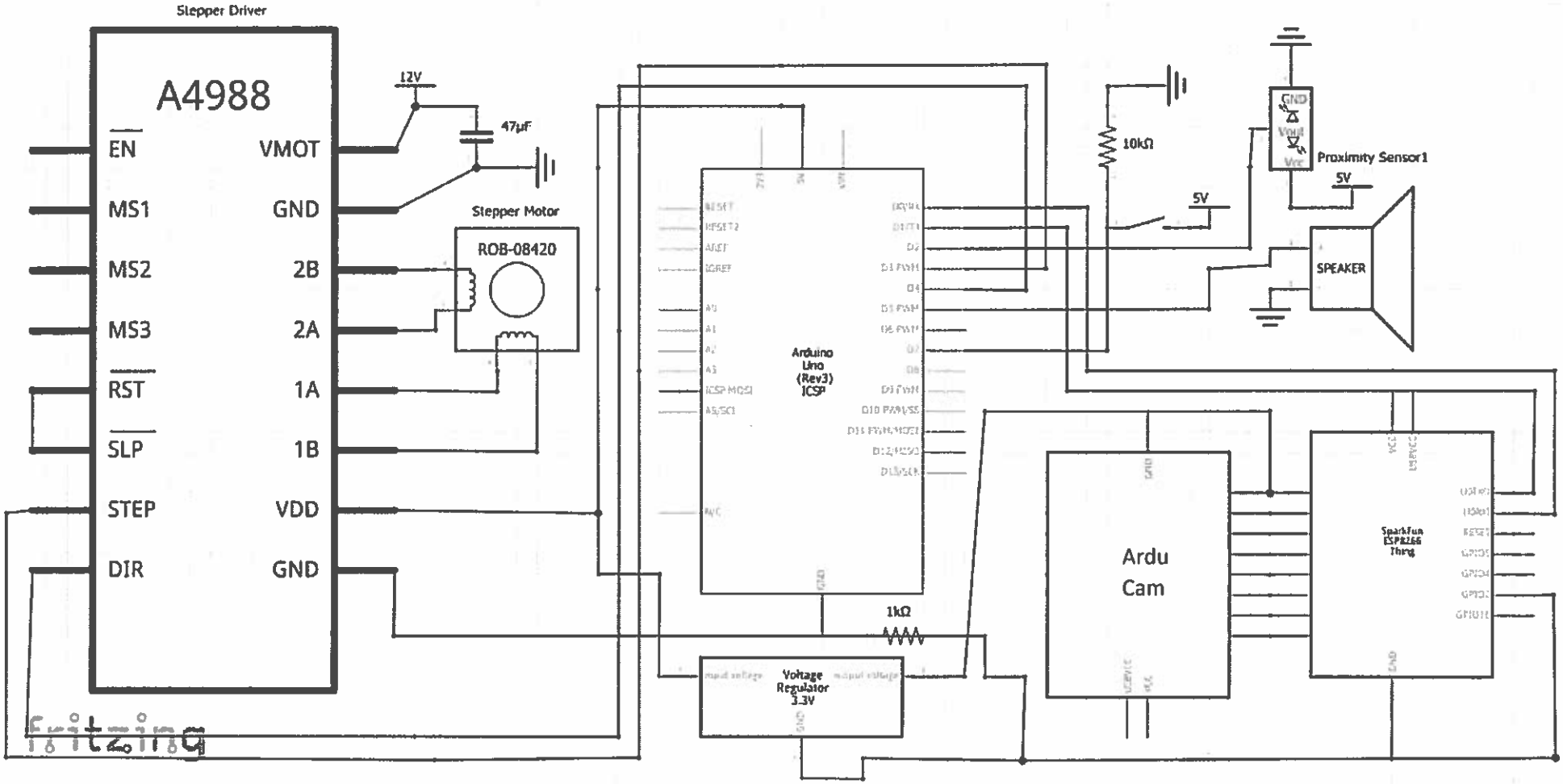
5. Focus on gradable content first and if you have time go back and add extra components to the project.

6. Ask other classmates (not in your group) for advice and help when stuck on something.

7. Look at online forums for pic and arduino troubleshooting.

Appendix A: Pic 16f88 wiring diagram




Stepper Driver
A4988
EN
MS1
MS2
MS3
RST
SLP
STEP
DIR
VMOT
GND
2B
2A
1A
1B
VDD
GND
12V
47µF
Stepper Motor
ROB-08420
Arduino
Uno
(Rev3)
ICSP
10kΩ
5V
GND
Vout
Vcc
Proximity Sensor1
5V
SPEAKER
1kΩ
Voltage
Regulator
3.3V
Ardu
Cam
Sparkfun
ESP8266
Thing
fritzing

Appendix C: Bill of Materials

	Item	Quantity	Total Price	Retailer
1	Stepper Motor	1	$13.50	Amazon
2	Motor Driver	1	$5.69	Amazon
3	Camera	1	$39.99	Amazon
4	ESP8266	1	$6.95	Sparkfun
5	PIR Sensor	1	$9.95	Sparkfun
6	Wire Jumpers	1 set of 10	$3.90	Sparkfun
7	Protoboard	1	$7.95	Sparkfun
8	Wire	5 Spools	$17.79	Sparkfun
9	Live Trap	1	$31.99	JAX Farm Supply
10	Push Button	1	$5.19	Home Depot
11	Heat Shrink	1 package	$3.98	Home Depot
12	Split Loom	1 coil	$4.89	Home Depot
13	9V Battery	2	$6.39	Batteries Plus
14	12V Battery	1	$13.95	Batteries Plus
15	Speaker	1	$4.75	Mountain State Electronics
16	Toggle Switch	2	$3.24	Radio Shack
17	9V Adaptor	1	$0.59	Mountain State Electronics
18	5 and 3.3V Voltage Regulators	2	$5.47	Mountain State Electronics
19	Arduino	1	$18.82	Amazon
20	Breadboard	2	$7.99	Amazon
21	LED's	2	$1.12	Amazon
22	PIC 16F88	1	Free	Microchip
23	M3 Fasteners	1 package	$2.19	Home Depot
24	3D Printer Fee	1	$25.00	I2P Lab
25	3d Printer Filament	1	$24.99	Amazon
26	JB Weld	1	$8.58	Ace Hardware
27	Gears	2	$12.98	Amazon
28	#4 Fasteners	2 packages	$2.53	Home Depot
		Grand Total	$290.36	

Appendix D: Pic16f88 code

```
' PIC16F88 code template for MECH307 Labs

' The following configuration bits and register settings
' enable the internal oscillator, set it to 8MHz,
' disables master clear, and turn off A/D conversion

' Configuration Bit Settings:
' Oscillator                          INTRC (INT102) (RA6 for I/O)
' Watchdog Timer                      Enabled
' Power-up Timer                      Enabled
' MCLR Pin Function                   Input Pin (RA5 for I/O)
' Brown-out Reset                     Enabled
' Low Voltage Programming             Disabled
' Flash Program Memory Write          Enabled
' CCP Multiplexed With                RB0
' Code                                Not Protected
' Data EEPROM                         Not Protected
' Fail-safe Clock Monitor             Enabled
' Internal External Switch Over       Enabled

' Define configuration settings (different from defaults)
#CONFIG
      __CONFIG _CONFIG1, _INTRC_IO & _PWRTE_ON & _MCLR_OFF & _LVP_OFF
#ENDCONFIG

' Set the internal oscillator frequency to 8 MHz
DEFINE OSC 8
OSCCON.4 = 1
OSCCON.5 = 1
OSCCON.6 = 1

' Turn off the analog to digital converters.

ansel = 0

' Put your code here:
```

```
' Declare variables and pin assignments
led0       Var         PORTB.0     'bit 0 green LED
led1       Var         PORTB.1     'bit 1 red LED

' Initialize the I/O pins
TRISA = %00001110      'PORTA pins 1,2,3 are inputs
TRISB = %00000000      'all PORTB pins are outputs

'Main Loop
Myloop:
      High PORTB.0                 'turn on green LED
      Low PORTB.1                  'turn off red LED

      Do While (PORTA.1 == 0)      'while switch is pressed
         Low PORTB.0               'turn off green LED
      High PORTB.1                 'turn on red LED
      Pause 100                    'pause 0.1 sec
      Low PORTB.1                  'turn off red LED
       Pause 100                   'pause 0.1 sec
      Loop


Goto myloop                        'go back to beginning of loop
```

Appendix E: Arduino Uno Code

```
#include "pitches.h"

// defines pins numbers
const int stepPin = 3;
const int dirPin = 4;
const int button = 7;
const int MOTION_PIN = 2; // Pin connected to motion detector
const int LED_PIN = 13; // used for the motion sensor
const int PIC = 8;
const int ESP = 6;

// define variables
int previous;
int reading;

// notes in the melody:
int melody[] = {
  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
  4, 8, 8, 4, 4, 4, 4, 4
};


void setup() {
  // Sets the two motor pins as outputs and the button pin as an input
  pinMode(stepPin,OUTPUT);
  pinMode(dirPin,OUTPUT);
  pinMode(button, INPUT);
  //calls the gate closed
  digitalWrite(dirPin,LOW);

    Serial.begin(9600);

  //sets up the motion sensor pins
```

```
 pinMode(MOTION_PIN, INPUT_PULLUP);
 pinMode(LED_PIN, OUTPUT);

// sets up the pic output and esp input
 pinMode(PIC, OUTPUT);
 digitalWrite(PIC, HIGH);
 pinMode(ESP, INPUT);

// gives reading and previous equal initial values
reading = LOW;
previous = LOW;
}


void loop() {

//sets up motion detector
 reading = digitalRead(button);
int proximity = digitalRead(MOTION_PIN);
 if (proximity == LOW) // If the sensor's output goes low, motion is detected
 {
  digitalWrite(LED_PIN, HIGH);
 }
 else
 {
  digitalWrite(LED_PIN, LOW);
 }

//when the button is pressed
if(((reading == HIGH && previous == LOW) || (reading == LOW && previous == HIGH)) ) {
 delay(100);
 if(((reading == HIGH && previous == LOW) || (reading == LOW && previous == HIGH)) ) {


 if(digitalRead(dirPin) == LOW) {  //if the gate is closed, it opens and delays for 5 seconds

 for(int x = 0; x < 95; x++) {
  digitalWrite(stepPin,HIGH);
  delayMicroseconds(500);
  digitalWrite(stepPin,LOW);
  delayMicroseconds(500);
 }
 delay(1000); // One second delay
```

```
 digitalWrite(dirPin,HIGH); //Changes the rotations direction
 // Makes 400 pulses for making two full cycle rotation
  digitalWrite(PIC, HIGH);
  delay(5000);
 }
else {  //if the gate is open, it closes
 for(int x = 0; x < 95; x++) {
   digitalWrite(stepPin,HIGH);
   delayMicroseconds(500);
   digitalWrite(stepPin,LOW);
         delayMicroseconds(500);

 }
 digitalWrite(dirPin,LOW); // Enables the motor to move in a particular direction
 // Makes 78 pulses for making 140.4 deg cycle rotation
 delay(1000);
         }
         previous = reading;
}
 }

//When a signal is sent from the ESP
if(digitalRead(ESP) == HIGH ) {
 delay(200);
 if(digitalRead(ESP)==HIGH) {

 if(digitalRead(dirPin) == LOW) {  //if the gate is closed, it opens and delays for 5 seconds

  for(int x = 0; x < 95; x++) {
    digitalWrite(stepPin,HIGH);
         delayMicroseconds(500);
    digitalWrite(stepPin,LOW);
    delayMicroseconds(500);
  }
  delay(1000); // One second delay
  digitalWrite(dirPin,HIGH); //Changes the rotations direction
  // Makes 400 pulses for making two full cycle rotation
   digitalWrite(PIC, HIGH);
   delay(5000);
  }
 else {                    //if the gate is open, it closes
  for(int x = 0; x < 95; x++) {
    digitalWrite(stepPin,HIGH);
```

```
    delayMicroseconds(500);
    digitalWrite(stepPin,LOW);
    delayMicroseconds(500);

  }
  digitalWrite(dirPin,LOW);
  delay(1000);
        }
}
}
```

if(digitalRead(LED_PIN) == HIGH && digitalRead(dirPin) == HIGH) { //if the motion sensor is triggered and the gate is open, the gate closes, the buzzer sound plays, and the light attached to the PIC flashes

```
  for(int x = 0; x < 95; x++) {
    digitalWrite(stepPin,HIGH);
    delayMicroseconds(500);
    digitalWrite(stepPin,LOW);
    delayMicroseconds(500);
  }
//PIC
  digitalWrite(PIC, LOW);

   // iterate over the notes of the melody:
 for (int thisNote = 0; thisNote < 8; thisNote++) {

        // to calculate the note duration, take one second
        // divided by the note type.
        //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
        int noteDuration = 1000 / noteDurations[thisNote];
        tone(9, melody[thisNote], noteDuration);

        // to distinguish the notes, set a minimum time between them.
        // the note's duration + 30% seems to work well:
        int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
        // stop the tone playing:
        noTone(9);

  }
  delay(1000); // One second delay
  digitalWrite(dirPin,LOW); //Changes the rotations direction
```

```
 // Makes 400 pulses for making two full cycle rotation
 }

}
```

Appendix F: ESP8266 Code

```
//libraries included
#include 'ESP8266WiFi.h'
#include 'WiFiClient.h'
#include 'ESP8266WebServer.h'
#include 'ESP8266mDNS.h'
#include 'Wire.h'
#include 'ArduCAM.h'
#include 'SPI.h'
#include 'memorysaver.h'
#if !(defined ESP8266 )
#error Please select the ArduCAM ESP8266 UNO board in the Tools/Board
#endif
//Error warning if the correct camera isn&#39;t connected
#if !(defined (OV2640_MINI_2MP)||defined (OV5640_MINI_5MP_PLUS) || defined
(OV5642_MINI_5MP_PLUS) \
|| defined (OV5642_MINI_5MP) || defined
(OV5642_MINI_5MP_BIT_ROTATION_FIXED) \
||(defined (ARDUCAM_SHIELD_V2) &amp;&amp; (defined (OV2640_CAM) || defined
(OV5640_CAM) || defined (OV5642_CAM))))
#error Please select the hardware platform and camera module in the
../libraries/ArduCAM/memorysaver.h file
#endif
//sets pin 2 to send send a pulse to the arduino
int PulsePin = 2;
// set GPIO16 as the slave select :
const int CS = 16;
//you can change the value of wifiType to select Station or AP mode.
//set the wifitype type to station
int wifiType = 0; // 0:Station 1:AP
//AP type settings
const char *AP_ssid = &quot;arducam_esp8266&quot;;
//Default is no password.If you want to set password,put your password here
const char *AP_password = &quot;&quot;;
//Station mode settings
const char *ssid = &quot;MIKE-PC789&quot;; // Put your SSID here
const char *password = &quot;67j422/C&quot;; // Put your PASSWORD here
static const size_t bufferSize = 1080;
```

```
static uint8_t buffer[bufferSize] = {0xFF};
uint8_t temp = 0, temp_last = 0;
int i = 0;
bool is_header = false;
//defining the camera
ESP8266WebServer server(80);
#if defined (OV2640_MINI_2MP) || defined (OV2640_CAM)
ArduCAM myCAM(OV2640, CS);
#elif defined (OV5640_MINI_5MP_PLUS) || defined (OV5640_CAM)
ArduCAM myCAM(OV5640, CS);
#elif defined (OV5642_MINI_5MP_PLUS) || defined (OV5642_MINI_5MP) || defined
(OV5642_MINI_5MP_BIT_ROTATION_FIXED) ||(defined (OV5642_CAM))
ArduCAM myCAM(OV5642, CS);
#endif
void start_capture(){
myCAM.clear_fifo_flag();
myCAM.start_capture();
}
//setting up the camera
void camCapture(ArduCAM myCAM){
WiFiClient client = server.client();
uint32_t len = myCAM.read_fifo_length();
if (len &gt;= MAX_FIFO_SIZE) //8M
{
Serial.println(F(&quot;Over size.&quot;));
}
if (len == 0 ) //0 kb
{
Serial.println(F(&quot;Size is 0.&quot;));
}
myCAM.CS_LOW();
myCAM.set_fifo_burst();
if (!client.connected()) return;
String response = &quot;HTTP/1.1 200 OK\r\n&quot;;
response += &quot;Content-Type: image/jpeg\r\n&quot;;
response += &quot;Content-len: &quot; + String(len) + &quot;\r\n\r\n&quot;;
server.sendContent(response);
i = 0;
while ( len-- )
{
temp_last = temp;
temp = SPI.transfer(0x00);
//Read JPEG data from FIFO
if ( (temp == 0xD9) &amp;&amp; (temp_last == 0xFF) ) //If find the end ,break while,
{
buffer[i++] = temp; //save the last 0XD9
//Write the remain bytes in the buffer
if (!client.connected()) break;
```

```
client.write(&amp;buffer[0], i);
is_header = false;
i = 0;
myCAM.CS_HIGH();
break;
}
if (is_header == true)
{
//Write image data to buffer if not full
if (i &lt; bufferSize)
buffer[i++] = temp;
else
{
//Write bufferSize bytes image data to file
if (!client.connected()) break;
client.write(&amp;buffer[0], bufferSize);
i = 0;
buffer[i++] = temp;
}
}
else if ((temp == 0xD8) &amp; (temp_last == 0xFF))
{
is_header = true;
buffer[i++] = temp_last;
buffer[i++] = temp;
}
}
}
void serverButton () { //setting up the button handle
WiFiClient client = server.client();

digitalWrite(PulsePin, HIGH);
delay(500);
digitalWrite(PulsePin, LOW);

// Read the first line of the request
String request = client.readStringUntil(&#39;\r&#39;);
Serial.println(request);
client.flush();

// Match the request

if (request.indexOf(&quot;/button&quot;) != -1) {
digitalWrite(PulsePin, HIGH);
delay(500);
digitalWrite(PulsePin, LOW);
}
```

```
// Return the response
client.println(&quot;HTTP/1.1 200 OK&quot;);
client.println(&quot;Content-Type: text/html&quot;);
client.println(&quot;&quot;); // do not forget this one
client.println(&quot;&lt;!DOCTYPE HTML&gt;&quot;);
client.println(&quot;&lt;html&gt;&quot;);

client.println(&quot;Gate Controls&quot;);
delay(1);
Serial.println(&quot;Client disonnected&quot;);
Serial.println(&quot;&quot;);

}
void serverCapture(){ //setting up the capture handle
start_capture();
Serial.println(F(&quot;CAM Capturing&quot;));
int total_time = 0;
total_time = millis();
while (!myCAM.get_bit(ARDUCHIP_TRIG, CAP_DONE_MASK));
total_time = millis() - total_time;
Serial.print(F(&quot;capture total_time used (in miliseconds):&quot;));
Serial.println(total_time, DEC);
total_time = 0;
Serial.println(F(&quot;CAM Capture Done.&quot;));
total_time = millis();
camCapture(myCAM);
total_time = millis() - total_time;
Serial.print(F(&quot;send total_time used (in miliseconds):&quot;));
Serial.println(total_time, DEC);
Serial.println(F(&quot;CAM send Done.&quot;));
}
void serverStream(){ //setting up the stream handle
WiFiClient client = server.client();
String response = &quot;HTTP/1.1 200 OK\r\n&quot;;
response += &quot;Content-Type: multipart/x-mixed- replace; boundary=frame\r\n\r\n&quot;;
server.sendContent(response);
while (1){
start_capture();
while (!myCAM.get_bit(ARDUCHIP_TRIG, CAP_DONE_MASK));
size_t len = myCAM.read_fifo_length();
if (len &gt;= MAX_FIFO_SIZE) //8M
{
Serial.println(F(&quot;Over size.&quot;));
continue;
}
if (len == 0 ) //0 kb
{
Serial.println(F(&quot;Size is 0.&quot;));
```

```
continue;
}
myCAM.CS_LOW();
myCAM.set_fifo_burst();
if (!client.connected()) break;
response = &amp;quot;-- frame\r\n&amp;quot;;
response += &amp;quot;Content-Type: image/jpeg\r\n\r\n&amp;quot;;
server.sendContent(response);
while ( len-- )
{
temp_last = temp;
temp = SPI.transfer(0x00);
//Read JPEG data from FIFO
if ( (temp == 0xD9) &amp;amp;&amp;amp; (temp_last == 0xFF) ) //If find the end ,break while,
{
buffer[i++] = temp; //save the last 0XD9
//Write the remain bytes in the buffer
myCAM.CS_HIGH();;
if (!client.connected()) break;
client.write(&amp;amp;buffer[0], i);
is_header = false;
i = 0;
}
if (is_header == true)
{
//Write image data to buffer if not full
if (i &amp;lt; bufferSize)
buffer[i++] = temp;
else
{
//Write bufferSize bytes image data to file
myCAM.CS_HIGH();
if (!client.connected()) break;
client.write(&amp;amp;buffer[0], bufferSize);
i = 0;
buffer[i++] = temp;
myCAM.CS_LOW();
myCAM.set_fifo_burst();
}
}
else if ((temp == 0xD8) &amp;amp; (temp_last == 0xFF))
{
is_header = true;
buffer[i++] = temp_last;
buffer[i++] = temp;
}
}
if (!client.connected()) break;
```

```
}
}
void handleNotFound(){ //setting up the IP display with no handle
String message = &amp;quot;Server is running!\n\n&amp;quot;;
message += &amp;quot;URI: &amp;quot;;
message += server.uri();
message += &amp;quot;\nMethod: &amp;quot;;
message += (server.method() == HTTP_GET)?&amp;quot;GET&amp;quot;:&amp;quot;POST&amp;quot;;
message += &amp;quot;\nArguments: &amp;quot;;
message += server.args();
message += &amp;quot;\n&amp;quot;;
server.send(200, &amp;quot;text/plain&amp;quot;, message);
if (server.hasArg(&amp;quot;ql&amp;quot;)){
int ql = server.arg(&amp;quot;ql&amp;quot;).toInt();
#if defined (OV2640_MINI_2MP) || defined (OV2640_CAM)
myCAM.OV2640_set_JPEG_size(ql);
#elif defined (OV5640_MINI_5MP_PLUS) || defined (OV5640_CAM)
myCAM.OV5640_set_JPEG_size(ql);
#elif defined (OV5642_MINI_5MP_PLUS) || defined
(OV5642_MINI_5MP_BIT_ROTATION_FIXED) ||(defined (OV5642_CAM))
myCAM.OV5642_set_JPEG_size(ql);
#endif
delay(1000);
Serial.println(&amp;quot;QL change to: &amp;quot; + server.arg(&amp;quot;ql&amp;quot;));
}
}
void setup() {
uint8_t vid, pid;
uint8_t temp;
#if defined(__SAM3X8E__)
Wire1.begin();
#else
Wire.begin();
#endif
Serial.begin(115200);
Serial.println(F(&amp;quot;ArduCAM Start!&amp;quot;));
// set the CS as an output:
pinMode(CS, OUTPUT);
// initialize SPI:
SPI.begin();
SPI.setFrequency(4000000); //4MHz
//Check if the ArduCAM SPI bus is OK
myCAM.write_reg(ARDUCHIP_TEST1, 0x55);
temp = myCAM.read_reg(ARDUCHIP_TEST1);
if (temp != 0x55){
Serial.println(F(&amp;quot;SPI1 interface Error!&amp;quot;));
while(1);
}
```

```
#if defined (OV2640_MINI_2MP) || defined (OV2640_CAM)
//Check if the camera module type is OV2640
myCAM.wrSensorReg8_8(0xff, 0x01);
myCAM.rdSensorReg8_8(OV2640_CHIPID_HIGH, &amp;amp;vid);
myCAM.rdSensorReg8_8(OV2640_CHIPID_LOW, &amp;amp;pid);
if ((vid != 0x26 ) &amp;amp;&amp;amp; (( pid != 0x41 ) || ( pid != 0x42 )))
Serial.println(F(&amp;quot;Can&amp;#39;t find OV2640 module!&amp;quot;));
else
Serial.println(F(&amp;quot;OV2640 detected.&amp;quot;));
#elif defined (OV5640_MINI_5MP_PLUS) || defined (OV5640_CAM)
//Check if the camera module type is OV5640
myCAM.wrSensorReg16_8(0xff, 0x01);
myCAM.rdSensorReg16_8(OV5640_CHIPID_HIGH, &amp;amp;vid);
myCAM.rdSensorReg16_8(OV5640_CHIPID_LOW, &amp;amp;pid);
if((vid != 0x56) || (pid != 0x40))
Serial.println(F(&amp;quot;Can&amp;#39;t find OV5640 module!&amp;quot;));
else
Serial.println(F(&amp;quot;OV5640 detected.&amp;quot;));
#elif defined (OV5642_MINI_5MP_PLUS) || defined (OV5642_MINI_5MP) || defined
(OV5642_MINI_5MP_BIT_ROTATION_FIXED) ||(defined (OV5642_CAM))
//Check if the camera module type is OV5642
myCAM.wrSensorReg16_8(0xff, 0x01);
myCAM.rdSensorReg16_8(OV5642_CHIPID_HIGH, &amp;amp;vid);
myCAM.rdSensorReg16_8(OV5642_CHIPID_LOW, &amp;amp;pid);
if((vid != 0x56) || (pid != 0x42)){
Serial.println(F(&amp;quot;Can&amp;#39;t find OV5642 module!&amp;quot;));
}
else
Serial.println(F(&amp;quot;OV5642 detected.&amp;quot;));
#endif
//Change to JPEG capture mode and initialize the OV2640 module
myCAM.set_format(JPEG);
myCAM.InitCAM();
#if defined (OV2640_MINI_2MP) || defined (OV2640_CAM)
myCAM.OV2640_set_JPEG_size(OV2640_320x240);
#elif defined (OV5640_MINI_5MP_PLUS) || defined (OV5640_CAM)
myCAM.write_reg(ARDUCHIP_TIM, VSYNC_LEVEL_MASK); //VSYNC is active HIGH
myCAM.OV5640_set_JPEG_size(OV5640_320x240);
#elif defined (OV5642_MINI_5MP_PLUS) || defined (OV5642_MINI_5MP) || defined
(OV5642_MINI_5MP_BIT_ROTATION_FIXED) ||(defined (OV5642_CAM))
myCAM.write_reg(ARDUCHIP_TIM, VSYNC_LEVEL_MASK); //VSYNC is active HIGH
myCAM.OV5640_set_JPEG_size(OV5642_2592x1944);
#endif
myCAM.clear_fifo_flag();
if (wifiType == 0){
if(!strcmp(ssid,&amp;quot;SSID&amp;quot;)){
Serial.println(F(&amp;quot;Please set your SSID&amp;quot;));
while(1);
```

```
}
if(!strcmp(password,&amp;quot;PASSWORD&amp;quot;)){
Serial.println(F(&amp;quot;Please set your PASSWORD&amp;quot;));
while(1);
}
// Connect to WiFi network
Serial.println();
Serial.println();
Serial.print(F(&amp;quot;Connecting to &amp;quot;));
Serial.println(ssid);
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(F(&amp;quot;.&amp;quot;));
}
Serial.println(F(&amp;quot;WiFi connected&amp;quot;));
Serial.println(&amp;quot;&amp;quot;);
Serial.println(WiFi.localIP());
}else if (wifiType == 1){
Serial.println();
Serial.println();
Serial.print(F(&amp;quot;Share AP: &amp;quot;));
Serial.println(AP_ssid);
Serial.print(F(&amp;quot;The password is: &amp;quot;));
Serial.println(AP_password);
WiFi.mode(WIFI_AP);
WiFi.softAP(AP_ssid, AP_password);
Serial.println(&amp;quot;&amp;quot;);
Serial.println(WiFi.softAPIP());
}
// Start the server
server.on(&amp;quot;/capture&amp;quot;, HTTP_GET, serverCapture);
server.on(&amp;quot;/stream&amp;quot;, HTTP_GET, serverStream);
server.on(&amp;quot;/button&amp;quot;, HTTP_GET, serverButton);
server.onNotFound(handleNotFound);
server.begin();
Serial.println(F(&amp;quot;Server started&amp;quot;));
pinMode(PulsePin, OUTPUT);
digitalWrite(PulsePin, LOW);
}
void loop() {
server.handleClient(); //connecting to esp to the IP with a certain handle
}
```