iKitty Smart Cat Feeder

ECE 110/120 Honors Lab

Haige Chen, Yaning Lan and Krystal Wu

Introduction

a. Statement of Purpose

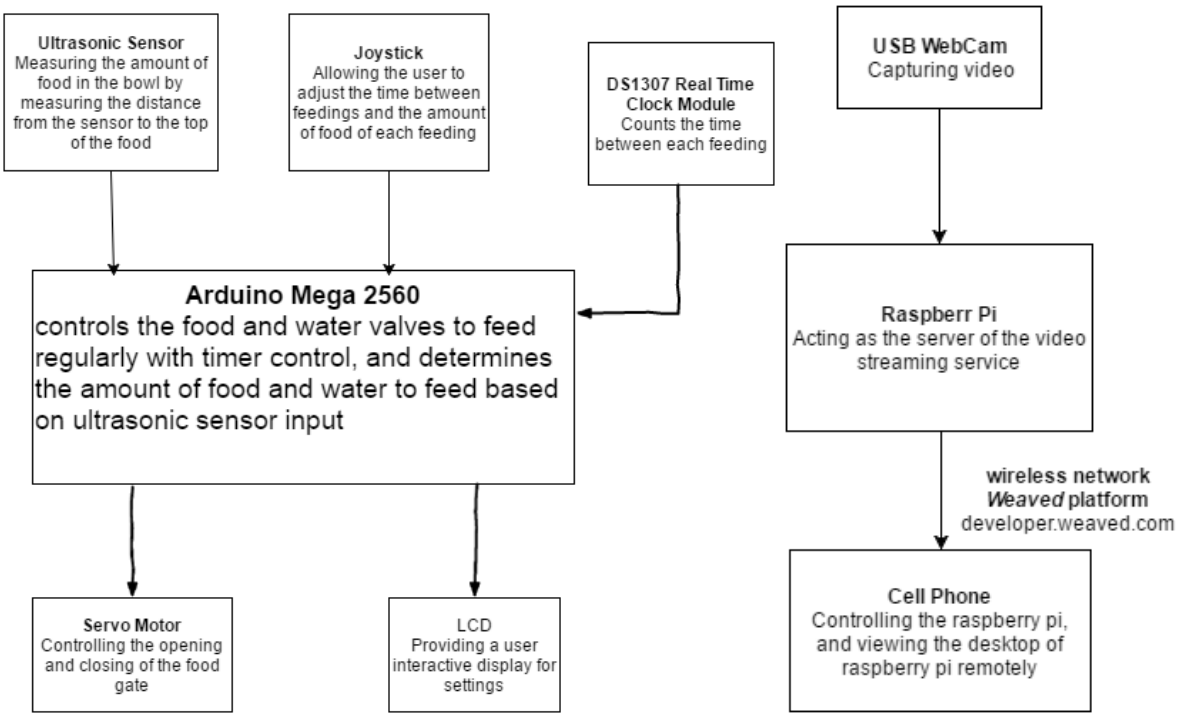
It is quite inconvenient to always have to have someone take care of your cat whenever you are out for a few days. While automatic cat feeders on amazon can certainly satisfy basic needs of a cat, we doubt if the small machine can truly be responsible for taking care of them when we are away from home. We want to build a machine that is able to feed the cat certain amount of food every certain number of hours, which can both be set by the owner. It will also have "facetime" functionality which will allow the owners to see their pets in real-time using their smartphones.

b. Features and Benefits

1. The user can change the settings so that the feeder will feed the desired amount of food over the desired time interval. (In the project, we did not complete the part for feeding water.)
2. The user can view the real-time video of their pets on their smartphones.

Design

a. System Overview



b. Design Detail

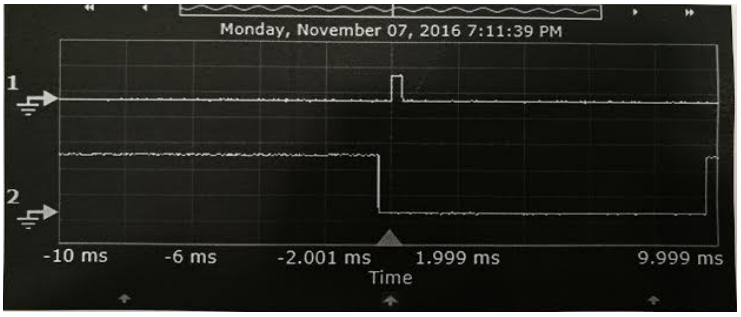
The project consists of two independent parts. In the first part, Arduino controls the feeding machine. The real time clock module is set so that every several hours, the Arduino will proceed its feeding sequence. In the feeding process, the Arduino controls the servo motor, which respectively control the opening and closing of food-feeding gate. The Arduino will close the gate if the ultrasonic sensor gives back values higher than the default setting so that the pets will not be overfed. Also, there is a joystick and an LCD screen allowing the user to set the time and amount of feeding accordingly.

In the second part, we want to establish an interface for the user to see and talk to their pets when he is away from home. The raspberry pi, and a webcam can be remotely accessed by user through wifi. The raspberry pi will be connected to the home wifi (or ethernet) and operate

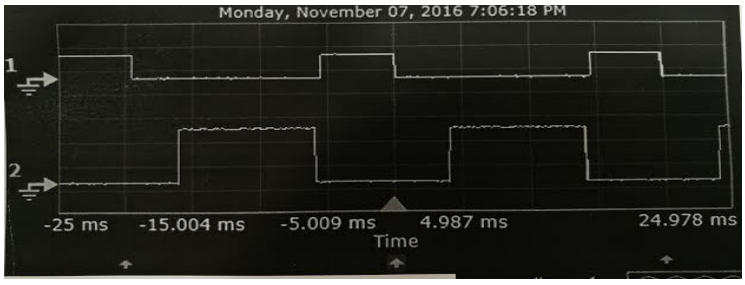
continuously. The user can use VNC client software on phones and PCs to remotely access raspberry pi's desktop in order to control the camera.

Result

We only used one sensor in this project — the ultrasonic sensor module. It can be used to measure the distance from a surface by measuring the time its sound signal takes to travel forth and back. In the diagrams below, the echo signal looks different for distant object and close object. The top line in each graph represents the echo signal. As soon as the trigger signal is emitted, the echo signal is pulled High, and will not become Low until it picks up the echo sound wave. By measuring the time it takes for the sound to travel, the distance can be determined.



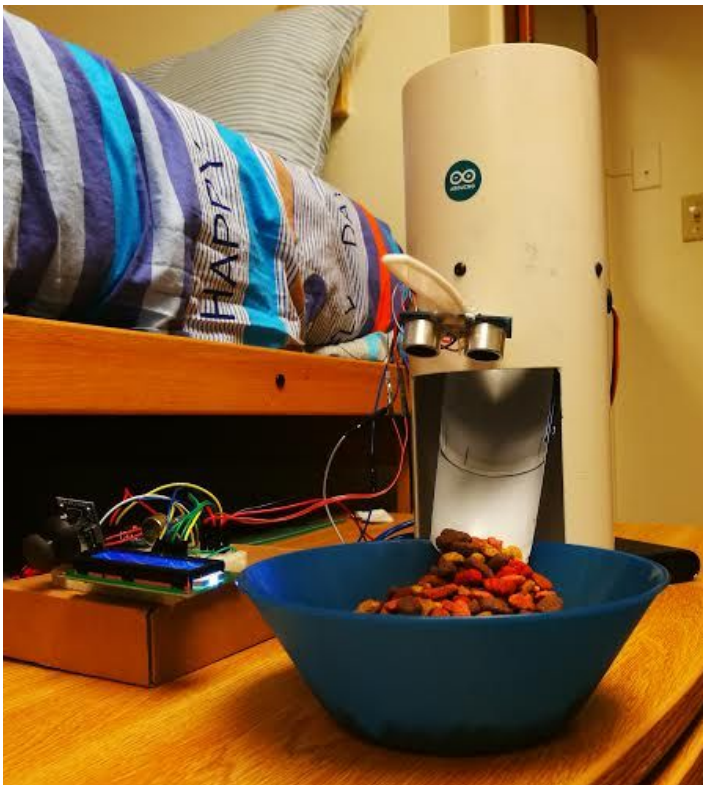
Object is 3 centimeters away



Object is 2 meters away

The final project was capable of executing most of the functions that it was designed to perform. For video demonstration, you can view the project at

https://www.facebook.com/andychenhaige/videos/691557791011467/. However, there were several unsolved problems. First, the ultrasonic sensor was still inconsistent when measuring from the surface of cat food. Second, the amount of food set by users hasn't been related to the actual feeding amount. Third, the water feeding part was not completed. Fourth, we still couldn't figure out how to transmit live audio stream so that the pet can hear its owner's voice.



Problems and Challenges

1. With the mechanical design, the food would not come out of the hole when it is opened. So we have to make the servo motor to rotate back and forth about the opening position to shake the food out.
2. Live transmitting the video was hard to implement. We solved it using a platform called *weaved* to share the raspberry pi desktop remotely.

3. The ultrasonic sensor is not good with measuring the distance from a surface of irregular shape. The top surface of the food is not flat. The reading from the ultrasonic sensor was often inconsistent. It is still an issue. It may be solved if we use another type of sensor.
4. The video streaming, although working, is not a fine, smooth video on the user's device. We are not sure what is the cause of the low frame rate. But we suspect that because the raspberry pi has its specially designed camera module can we were simply using a webcam, the compatibility may affect the frame rate.

Future Plans

We would like to work on the functionality that has not been accomplished to make it actually marketable. We also would like to make it more interesting by using more functions of the raspberry pi. We can use the raspberry pi as a server to control the circuit, and allow the user to access the settings via text.

References

[1]"Build a Raspberry Pi Webcam Server using Motion - Scott on Technology", *Scott on Technology*, 2016. [Online]. Available: https://scottontechnology.com/raspberry-pi-webcam-server-using-motion/. [Accessed: 12- Dec- 2016].

[2]"Setup Weaved and the Raspberry Pi", *Instructables.com*, 2016. [Online]. Available: http://www.instructables.com/id/Setup-Weaved-and-the-Raspberry-Pi/?ALLSTEPS. [Accessed: 12- Dec- 2016].

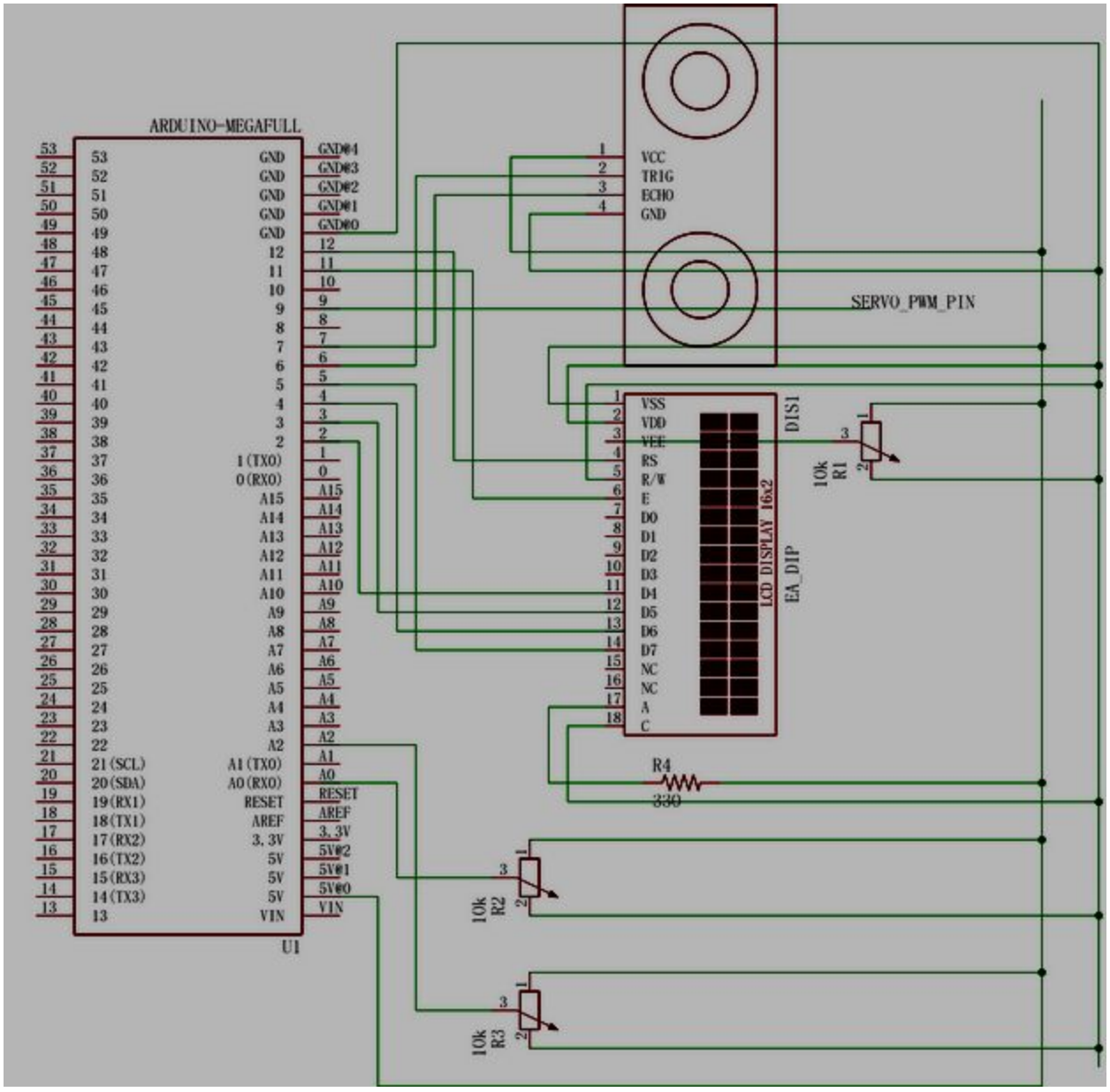
[3]"Arduino - LiquidCrystalDisplay", *Arduino.cc*, 2016. [Online]. Available: https://www.arduino.cc/en/Tutorial/LiquidCrystalDisplay. [Accessed: 12- Dec- 2016].

[4]"Arduino Playground - NewPing Library", *Playground.arduino.cc*, 2016. [Online]. Available: http://playground.arduino.cc/Code/NewPing. [Accessed: 12- Dec- 2016].

[5]"Arduino - Servo", *Arduino.cc*, 2016. [Online]. Available: https://www.arduino.cc/en/Reference/Servo. [Accessed: 12- Dec- 2016].

Appendix

Schematics



Code

```
#include <NewPing.h>
```

```
#define TRIGGER_PIN  6

#define ECHO_PIN     7

#define MAX_DISTANCE 200

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

#include <LiquidCrystal.h>

#include <Time.h>

#include <Wire.h>

#include <DS1307RTC.h>

#include <Servo.h>

 Servo myservo;

int pos = 0;

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

int feedingAmt=400;

int feedingTime=7;

int mode=0;

int delayTime=150;

float x=2.5, y=2.5;

String time;

int t=7;  //t is how many seconds between each feeding

void setup()

{

  lcd.begin(16, 2);
```

```
myservo.attach(9);
myservo.write(0);
delay(300);
myservo.detach();
setTime(0,0,0,1,1,1970);
lcd.print("Welcome to iKitty");
delay(2000);
mode=0;

while(1)
{
 motor();
 if(mode==0)
 {
 page1();
 //lcd.setCursor(15,1);
 lcd.noBlink();
 while(mode==0)
 {
  mode01();
  motor();
 }
```

```
}
if(mode==1)
{
page1();
lcd.setCursor(11,0);
lcd.blink();
int i=0;
while(mode==1)
{
 i++;
 motor();
 mode12();
 amtUp();
 amtDown();
 if(i>=400)
 {
 i=0;
 page1();
 lcd.setCursor(11,0);
 lcd.blink();
 }
}
```

```
}


if(mode==2)
{
lcd.setCursor(9,1);
lcd.blink();
int i=0;
while(mode==2)
{
  motor();
mode23();
timeUp();
timeDown();
i++;
if(i>=400)
  {
  i=0;
  page1();
  lcd.setCursor(9,1);
  lcd.blink();
  }
}
```

```
    }

    if(mode==3)
    {
    lcd.clear();
    page2();
    while(mode==3)
    {
      motor();
      page2();
      mode30();
    }
    }
}
}

void loop()
{
}

void page1()
{
```

```
  String line1=String("Set Amount ") + feedingAmt + String(" g");
  String line2=String("Set time ") + feedingTime + String(" h");
  lcd.setCursor(0,0);
  lcd.print(line1);
  lcd.setCursor(0,1);
  lcd.print(line2);
}

void page2()
{
  time=String(now());
  lcd.setCursor(0,0);
  lcd.noBlink();
  lcd.print(time+" ");
}

void readVoltage()
{
x=analogRead(0) * (5.0 / 1023.0);
y=analogRead(2) * (5.0 / 1023.0);
}
```

```
void mode01()
{
  readVoltage();
  if (y>=4.5)
  {
    delay(delayTime);
    readVoltage();
    if (y>=4.5)
    {
    mode=1;
    }
  }
}
void mode12()
{
  readVoltage();
  if (y>=4.5)
  {
    delay(delayTime);
    readVoltage();
    if (y>=4.5)
    {
```

```
    mode=2;
    }
  }
}

void mode23()
{
  readVoltage();
  if (y>=4.5)
  {
    delay(delayTime);
    readVoltage();
    if (y>=4.5)
    {
    mode=3;
    }
  }
}

void mode30()
{
  readVoltage();
```

```
  if (y>=4.5)
  {
    delay(delayTime);
    readVoltage();
    if (y>=4.5)
    {
    mode=0;
    }
  }
}

void amtDown()
{
  readVoltage();
  if (x<=0.5)
  {
    delay(delayTime);
    readVoltage();
    if (x<=0.5)
    {
    feedingAmt=feedingAmt-30;
    }
```

```
  }
}

void amtUp()
{
  readVoltage();
  if (x>=4.5)
  {
    delay(delayTime);
    readVoltage();
    if (x>=4.5)
    {
    feedingAmt=feedingAmt+30;
    }
  }
}

void timeDown()
{
  readVoltage();
  if (x<=0.5)
  {
```

```
    delay(delayTime);
    readVoltage();
    if (x<=0.5)
    {
    feedingTime=feedingTime-1;
    }
  }
}

void timeUp()
{
  readVoltage();
  if (x>=4.5)
  {
    delay(delayTime);
    readVoltage();
    if (x>=4.5)
    {
    feedingTime=feedingTime+1;
    }
  }
}
```

```
double distance()
{
  double mSec=sonar.ping();
  delay(5);
  double mm=mSec/2.9411/2;
  return mm;
}

void motor()
{
if (now()>feedingTime)
{
  if(sonar.convert_cm(sonar.ping_median(4))>=13.5)
  {
  myservo.attach(9);
  for(pos = 0; pos < 145; pos += 1)  // goes from 0 degrees to 180 degrees
  {                                  // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(5);                        // waits 15ms for the servo to reach the position
  }
  myservo.detach();
```

```
digitalWrite(9,LOW);
int i=0;
double dis=sonar.convert_cm(sonar.ping_median(4));
while(dis>12)
{
i++;
dis=sonar.convert_cm(sonar.ping_median(4));
myservo.attach(9);
for(pos = 145; pos>=110; pos-=1)     // goes from 180 degrees to 0 degrees
{
  myservo.write(pos);              // tell servo to go to position in variable 'pos'
  delay(5);                       // waits 15ms for the servo to reach the position
}
delay(5);
for(pos = 110; pos < 145; pos += 1)  // goes from 0 degrees to 180 degrees
{                                  // in steps of 1 degree
  myservo.write(pos);              // tell servo to go to position in variable 'pos'
  delay(5);                       // waits 15ms for the servo to reach the position
}
}
myservo.attach(9);
for(pos = 145; pos>=1; pos-=1)     // goes from 180 degrees to 0 degrees
```

```
  {
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(5);                        // waits 15ms for the servo to reach the position
  }
  }
   setTime(0,0,0,1,1,1970);
   myservo.detach();
  digitalWrite(9,LOW);
}
}
```