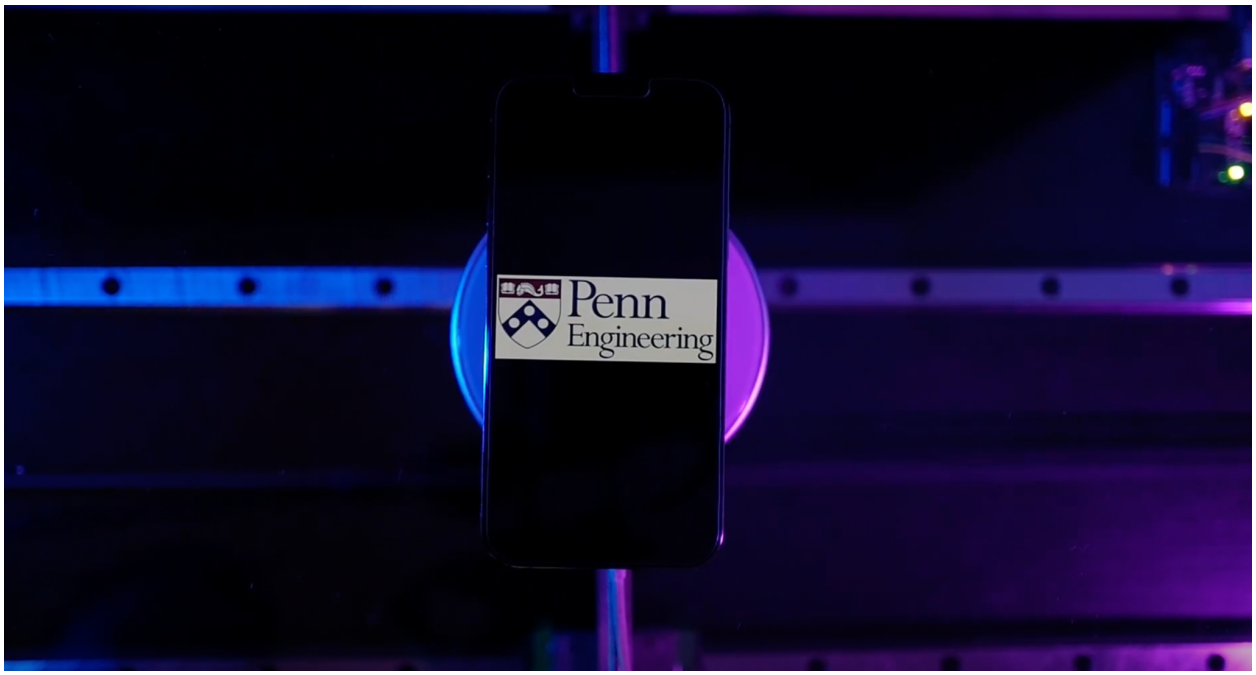
[THE DESK]



Ziyi Yang
Keran Wang

https://devpost.com/software/the-desk

https://github.com/Zakiyanggg/TheDesk

Final Project, Fall 2021
ESE519/IPD519: Real-Time and Embedded Systems
University of Pennsylvania

Table of Contents

Abstract

We decided to design a system that can automatically locate your phone on a surface and the wireless charging base will move to the right position and start charging your phone or other wireless charging device. This could be applied to a bed side table or a working desk. My initial idea is using Raspberry Pi to support the CV part and generate coordination information to the motor control system, which I prefer to use ATmega328p MCU.

For sensors I just need a webcam to capture the object on the table surface. I might upgrade the camera to stereo type in order to get more accurate location information. I need two step motors to drive the linear slide rails. Two sets of linear rails can let the charging pad move across the entire surface. A charging pad that supports most common fast charging devices. It might be hard to train the ML module to recognize a smart phone on a messy table and Raspberry pi only supports TensorFlow Lite which the functionality is quite limited. I also need to design a system to collect the free wires flying around two rails, to make sure they won't entangle. For safety features, I need to put switches that prevent the motors going too far that might damage the rails or gears.

Motivation

Wireless charging has been widely used on digital devices. Users can just simply put their phone on the charging pad and pick it up when needed without unplugging any cables. But wireless charging has some limitations, the device has to be put right at the center of the charging pad, otherwise the two coils would not be able to transfer energy. To make our life much easier, we decided to design a desk that the whole surface can act like a charging pad, so users don't have to worry about where to put the phone. And the same method can be scaled and applied to a bed side table or other surfaces to match any design requirement.

Goals

- Milestone 1
  - **Two linear rails can drive the wireless charger to destination**
  - **Two motors can move simultaneously**
  - **Mount the camera in right location which included the surface of the desk**
  - **Test wireless charging pad and then ensure it can charge through glass panel**
- Final Demo
  - **The converting coordinate system correctly represents the device location**
  - **The Computer Vision module different devices on the surface**
  - **The pad can move to the phone's converting coordinates and successfully charging the phone**
  - **After fully charging first device, the charging pad can move to the next device**

Methodology

Camera + Pi

➜ Use Computer Vision to locate the device

Ran OpenCV on raspberry pi with coco database(More detailed information in Appendix A) to achieve locating phones on the desk surface.

openCV view

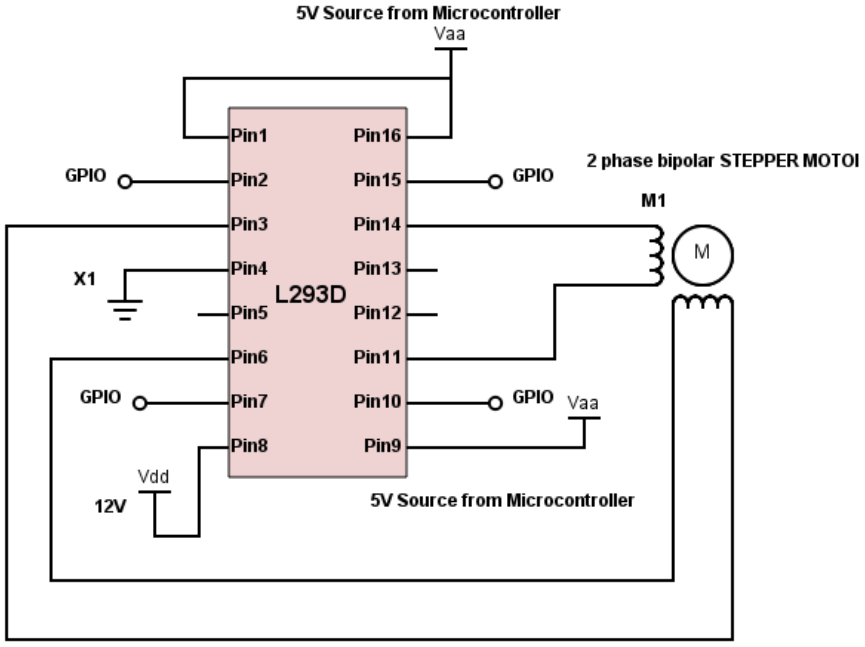
➜ Convert the location information to the preset coordinate system

Trained coco dataset with openCV. Code attached in Appendix A. The official detection model from the COCO dataset could detect phones but in our case, we need a model that provides more accurate detection so we took 150 photos of iphone in different angles and light conditions. Which as shown in the image above, could successfully detect multiple phones on the desk surface.

➜ Drive the step motors

At first, we bought a V1 motor shield for the arduino board, but then we figured out it failed to drive two NEMA 17 stepper motors even with extra power supply enabled. However, by learning from the board, we found out the L293D is the chip that we were looking for, so we directly took it from the board and mounted it on the breadboard which gave us more flexibility and more functionality compared to the original setup.

We used two ATmega328p MCUs to drive two stepper motors. And two L293D motor control chips. The L293D is an IC with two H-Bridges, and each H-Bridge will drive one of the electromagnetic coils of a stepper motor. By energizing these electromagnetic coils in a specific sequence, the shaft of a stepper can be moved forward or backward precisely in small steps. Before we start hooking the motor up with the chip, you will need to determine the A+, A-, B+ and B- wires on the motor you plan to use. The best way to do this is to check the datasheet of the motor. For our motor these are red, green, blue and yellow.



We tried a lot of different speed tests for our motors since we changed one of the Linear slides to Belt driven system, so the ratio of moving is changed by the step size of the end wheel on the both ends of the belt.

Another extra problem we encountered was the same ground was shared between the two boards. Though we gained the benefits

of better stability for driving two stepper motors at the same time, as we used our hand to scroll the rail to move the mounting block to the initial position, we figured out that the other arduino board's LED indicator lighted up even without the power being turned on.

As a result, we tried different approaches to avoid this problem because it will not only interface our ADC reading but also will make the other motor move weirdly. The approaches included using capacitor, transistor bridge, huge metal block or diodes on both VCCs and GNDs. In the end, we found with the limited electronics parts we had the best way is to use the diode on the shared ground for better space management and less complexity when we need to debug the circuits.

Linear Slides/Belt + Step motor

➜ Create two linear slides system

We used two sets of linear sliders/rails. The horizontal rail is built by two linear slides and a belt that's driven by a stepper motor. The vertical rail is a linear screw rail. The two slides/rail can go through the entire surface of the desk.

➜ Set start / stop point on slides

For this project, we still have to manually adjust the charging pad at the center of the table each time it restarts. We tried to add self alignment into our project so that it can automatically go to an initial position. But that function caused us a lot of problems since the stepper motors are not stable, sometimes it just randomly runs out of the rail and all positioning information had to be reset.

➜ Push charging pad to the desired location

The method we use to let the pi and uno board communicate with

each other is, we first enabled GPIO pins on the pi and when it get the location of the phone, it computes the distance that the charging pad has to travel in X/Y direction and convert it into a short HIGH signal on corresponding pins and let ATmega328p to measure how long is that HIGH signal then converts it to steps for each motor to travel. Since the output voltage of the pi is 3.3v we had to use a logic level shifter to get a 5v signal.

➜ Reset after charging finished

We achieved this part with a tricky method. The charging pad we used for this project was bought from amazon and it has LED lights that indicate the charging status. When it's green, it means the charging is in process and when it finishes charging, it turns off. So we wired that LED to the pi's GPIO pins and let pi to detect voltage change on the LED. So that our code would know if one device has been fully charged or removed from the charging pad.

Results

Our project could successfully locate phones on the table and drive the charging pad to the desired location to charge the phones. When pi detects a phone on the desk, it starts counting how long the phone remains at the same spot. Over 3s without significant displacement, the Pi will compute the X/Y displacement for the charging pad to move and send HIGH states to corresponding pins. The ATmega328p receives HIGH and triggol the timer to count how long the HIGH signal is, then convert the signal into steps to drive the motors.

We successfully achieved all the topics on the 'goals' part. Though we got a lot of comments from the proposal part about if we will finish the project with such an amount of time due to the

complexity, the result came out better than what we expected from the beginning because the detecting speed from the computer vision module and the moving speed from the linear rails/belt system are more precise and faster than we thought.

Conclusion

During the project, we got more familiar with ATmega328p development and tried to let different MCUs communicate with each other. The process of generating object detection was a pain but the result turned out good enough to achieve our need.

We learnt the basics behind object detection, how stepper motors work and how precise the linear screw rails could achieve. The belt based rail could achieve our needs but compared to the screw rail, it sometimes stuck and one or two steps would be missing, in our case, it's not a big deal but if we bring it to a smaller scale, precision becomes more important. We firstly decided to use two screw rails, but a 1000mm screw rod with specific screw pitch lead is so pricey and takes a long time to make.

For a more smooth movement of the charging pad, we could add two more stepper motors and place the 4 motors on each corner so there won't be a motor moving around the table, the height can be reduced down about 5-10cm.

The next step could be adding IoT functionalities into our project. For example, a real time weather display and ToDo list that can be edited by the user via phone. User can interact with the desk by hand signs or touch.

References

Object detection:
https://cocodataset.org/#download –COCO dataset
https://www.youtube.com/watch?v=HXDD7-EnGBY –Object detection on Pi

Last edited on 2020/08/02 18:18 EDT

https://towardsdatascience.com/how-to-work-with-object-detection-datasets-in-coco-format-9bf4fb5848a4 –Train COCO detection model

Stepper motor:
https://www.youtube.com/watch?v=rE6ZgBLFCFw –L293D stepper motor driver

Appendix A

Our code for ATmega328p:

Last edited on 2020/08/02 18:18 EDT

```c
/* * TheDesk.c
 * This program is for ESE519 Final Project
 * Group 44
 * The Desk
 * Author : Ziyi Yang, Keran Wang
 */


#define F_CPU 16000000UL
#include <avr/io.h>
#include <time.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <util/delay.h>
#define BAUD_RATE 9600
#define BAUD_PRESCALER (((F_CPU / (BAUD_RATE * 16UL))) - 1)
#include "uart.h"
#include <avr/interrupt.h>

int step_delay;
int last_step_time = 0;
int step_number = 0;
int number_of_steps;

int Timer1_ovf;
int temp;
int temp1;
int count;
char String[25];

//function sets steps for stepper to run the sequence
//The sequence of control signals for 4 control wires is as follows:
/*
* Step C0 C1 C2 C3
*    1  1  0  1  0
*    2  0  1  1  0
*    3  0  1  0  1
*    4  1  0  0  1
*/
```

```
void stepMotorForward(int step)
{
    switch(step){
        case 0:
        PORTB |= (1<<PORTB1);//pb1 high
        PORTB &= ~(1<<PORTB2);//pb2 low
        PORTB |= (1<<PORTB3);//pb3 high
        PORTB &= ~(1<<PORTB4);//pb4 low
        case 1:
        PORTB &= ~(1<<PORTB1);//pb1 low
        PORTB |= (1<<PORTB2);//pb2 high
        PORTB |= (1<<PORTB3);//pb3 high
        PORTB &= ~(1<<PORTB4);//pb4 low
        case 2:
        PORTB &= ~(1<<PORTB1);//pb1 low
        PORTB |= (1<<PORTB2);//pb2 high
        PORTB &= ~(1<<PORTB3);//pb3 low
        PORTB |= (1<<PORTB4);//pb4 high
        case 3:
        PORTB |= (1<<PORTB1);//pb1 high
        PORTB &= ~(1<<PORTB2);//pb2 low
        PORTB &= ~(1<<PORTB3);//pb3 low
        PORTB |= (1<<PORTB4);//pb4 high
    }
}

void stepMotorBackward(int step)
{
    switch(step){
        case 0:
        PORTB |= (1<<PORTB1);//pb1 high
        PORTB &= ~(1<<PORTB2);//pb2 low
        PORTB |= (1<<PORTB3);//pb3 high
        PORTB &= ~(1<<PORTB4);//pb4 low
        case 1:
        PORTB |= (1<<PORTB1);//pb1 high
        PORTB &= ~(1<<PORTB2);//pb2 low
        PORTB &= ~(1<<PORTB3);//pb3 low
        PORTB |= (1<<PORTB4);//pb4 high
        case 2:
        PORTB &= ~(1<<PORTB1);//pb1 low
        PORTB |= (1<<PORTB2);//pb2 high
        PORTB &= ~(1<<PORTB3);//pb3 low
        PORTB |= (1<<PORTB4);//pb4 high
        case 3:
        PORTB &= ~(1<<PORTB1);//pb1 low
        PORTB |= (1<<PORTB2);//pb2 high
        PORTB |= (1<<PORTB3);//pb3 high
        PORTB &= ~(1<<PORTB4);//pb4 low
    }
}
```

```
void stepForward(int step)
{
    int step_left = step;
    while(step_left >0){
        time_t now = time(0);
        if(now -last_step_time >= step_delay){
            last_step_time = now;
            step_number++;
            if(step_number == number_of_steps){
                step_number = 0;
            }
            stepMotorForward(step_number%4);
        }
    }
}

void stepBackward(int step)
{
    int step_left = step;
    while(step_left >0){
        time_t now = time(0);
        if(now -last_step_time >= step_delay){
            last_step_time = now;
            step_number++;
            if(step_number == number_of_steps){
                step_number = 0;
            }
            stepMotorBackward(step_number%4);
        }
    }
}

ISR(TIMER1_OVF_vect){
    Timer1_ovf++;//overflow every 0.004100s-->4100us
}
```

```
void Initialize()
{
    cli();
    //set 4 output pins to drive 4pin stepper motor
    DDRB |= (1 << DDB1);//PB1 as output
    DDRB |= (1 << DDB2);//PB2 as output
    DDRB |= (1 << DDB3);//PB3 as output
    DDRB |= (1 << DDB4);//PB4 as output
    DDRD &= ~(1<< DDD5);//PD5 as input
    DDRD &= ~(1<< DDD7);//PD7 as input
    TIMSK1 |= (1<<TOIE1);//Enable timer1 ovf
    int forward;
    int backward;
    int motorstep;

    sei();
    UART_init(BAUD_PRESCALER);
}
int main(void)
{
    Initialize();
    while (1)
    {
        if(PIND & (1<<PIND5)){
        TCCR1B |= (1<<CS10);//start timer1 with prescaler = 1
        sprintf(String,"PD5 HIGH,forward");
        UART_putstring(String);
        temp = 0 ;
    }
    }
    while(temp ==0){
    count++;
        if(PIND & ~(1<<PIND5)){
            sprintf(String,count);
            UART_putstring(String);
            count = Timer1_ovf/4;
            stepMotorForward(count);
            stepForward(count);
            temp = 1;
            TCCR1B &= ~(1<<CS10);//stop timer1
            Timer1_ovf = 0;
        }
    }
    if(PIND & (1<<PIND7)){
        TCCR1B |= (1<<CS10);//start timer1 with prescaler = 1
        sprintf(String,"PD7 HIGH,backward");
        UART_putstring(String);
        temp = 0;
    }
```

```
    while(temp ==0){
        count++;
        if(PIND & ~(1<<PIND7)){
            sprintf(String,count);
            UART_putstring(String);
            count = Timer1_ovf/4;
            stepMotorBackward(count);
            stepBackward(count);
            temp = 1;
            TCCR1B &= ~(1<<CS10);//stop timer1
            Timer1_ovf = 0;
        }
    }
}
```

Code for phone detection:

```
import cv2
import RPi.GPIO as GPIO
import time

pin1  = 16#Y-
pin2 = 18#18 Y+

pin3 = 24 #X+
pin4 = 22#A1 X-

Xpos1 = 680
Xpos2 = 680
Ypos1 = 330
Ypos2 = 3390

fcount = 0
Xdiff = 0
Ydiff = 0
center1 = 0
center2 = 0

GPIO.setmode(GPIO.BOARD)
GPIO.setup(pin1,GPIO.OUT)
GPIO.setup(pin2,GPIO.OUT)
GPIO.setup(pin3,GPIO.OUT)
GPIO.setup(pin4,GPIO.OUT)

classNames = []
classFile = "/home/pi/Desktop/ESE519Final/Object-detection/coco/Object_Detection_Files/coco.names"
with open(classFile,"rt") as f:
    classNames = f.read().rstrip("\n").split("\n")

configPath = "/home/pi/Desktop/ESE519Final/Object-detection/coco/Object_Detection_Files/ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt"
weightsPath = "/home/pi/Desktop/ESE519Final/Object-detection/coco/Object_Detection_Files/frozen_inference_graph.pb"

net = cv2.dnn_DetectionModel(weightsPath,configPath)
net.setInputSize(320,320)
net.setInputScale(1.0/ 127.5)
net.setInputMean((127.5, 127.5, 127.5))
net.setInputSwapRB(True)

```

Last edited on 2020/08/02 18:18 EDT

```
def getObjects(center1,center2,img, thres, nms, draw=True, objects=[]):
    classIds, confs, bbox = net.detect(img,confThreshold=thres,nmsThreshold=nms)
    #print(classIds,bbox)
    #cv2.circle(img,(960,540),2,(255,0,0),2)
    if len(objects) == 0: objects = classNames
    objectInfo =[]
    if len(classIds) != 0:
        for classId, confidence,box in zip(classIds.flatten(),confs.flatten(),bbox):
            className = classNames[classId - 1]
            if className in objects:
                objectInfo.append([box,className])
                if (draw):
                    #print(box)
                    center1 = int((box[0]+box[0]+box[2])/2)
                    center2 = int((box[1]+box[1]+box[3])/2)-20

                    #print(center1,center2)
                    #print(box)
                    #print("The coordinate of the phone is at\n")
                    #print(center1,center2-30,round(confidence*100))
                    #getPos(center1,center2)
                    # yhnhgb cv2.circle(img,(center1,center2),100,(0,0,255),5)
                    cv2.circle(img,(center1,center2),int(0.8*box[2]/2),(0,0,255),5)
                    #cv2.circle(img,(box[0]+box[2],box[1]+box[3]),5,(0,0,255),5)

                    cv2.rectangle(img,box,color=(0,255,0),thickness=2)
                    cv2.putText(img,classNames[classId-1].upper(),(box[0]+10,box[1]+30),
                    cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),2)
                    cv2.putText(img,str(round(confidence*100,2)),(box[0]+220,box[1]+30),
                    cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),2)

    return img,objectInfo,center1,center2


```

```
if __name__ == "__main__":

    cap = cv2.VideoCapture(0)
    cap.set(3,1280)
    cap.set(4,720)
    #cap.set(10,70)
    #starttime =time.time()

    while True:

        success, img = cap.read()
        #cv2.circle(img,(0,0),5,(0,0,255),5)
        #cv2.circle(img,(1920,1080),5,(0,0,255),5)
        #nowtime = time.time()
        #if(int(nowtime-starttime)>2):
        result, objectInfo, Xpos2, Ypos2= getObjects(center1,center2,img,0.3,0.3,objects=['cell phone'])
        if(abs(Xpos2-Xpos1)>=10 or abs(Ypos2-Ypos1)>=10):
            fcount = fcount+1
        if(abs(Xpos2-Xpos1)<10 and abs(Ypos2-Ypos1)<10):
            fcount = 0
        if(Xpos2 == 0):
            fcount = 0
        if(fcount == 3):
            stepY = (120/196)*23.78*(abs(Ypos2-Ypos1))
            stepX = (363/493)*(2200/570)*(abs(Xpos2-Xpos1))
            Yout = (1/4400)*stepY
            Xout = (1/2200)*stepX
            print("Target Locked!")
            if(Xpos2>=Xpos1):#X+
                GPIO.output(pin4,GPIO.HIGH)
                time.sleep(Xout)
                GPIO.output(pin4,GPIO.LOW)
                time.sleep(1)
            if(Xpos2<Xpos1):#X-
                GPIO.output(pin3,GPIO.HIGH)
                time.sleep(Xout)
                GPIO.output(pin3,GPIO.LOW)
                time.sleep(1)
            if(Ypos2>=Ypos1):#Y+
                GPIO.output(pin2,GPIO.HIGH)
```

```
134                     time.sleep(Yout)
135                     GPIO.output(pin2,GPIO.LOW)
136                     time.sleep(1)
137                 if(Ypos2<Ypos1):#Y-
138                     GPIO.output(pin1,GPIO.HIGH)
139                     time.sleep(Yout)
140                     GPIO.output(pin1,GPIO.LOW)
141                     time.sleep(1)
142                 Ypos1 = Ypos2
143                 Xpos1 = Xpos2
144                 #starttime = time.time()
145             #print("Location:")
146             #print(Xpos2,Ypos2-30)
147             cv2.imshow("Output",img)
148             print(fcount)
149             print(Xpos2,Ypos2)
150             if cv2.waitKey(1) ==27:#press ESC to stop the code
151                break
152         cap.release()
153         cv2.destroyAllWindows()
```