# AI CASHIER

Weinian Feng 6338226421
Pattadon Naksuwan 63381532221
Paranut Prasittipap 6338139521
Suphakrit Kotchasarn 6338217821




# Outline

1. Inspiration and Overview
2. Our components and assemblies
3. Our journey
4. Final product
5. Challenges and lesson learnt
6. Furture improvements

# Inspiration

imagine you are the cashier at a mall

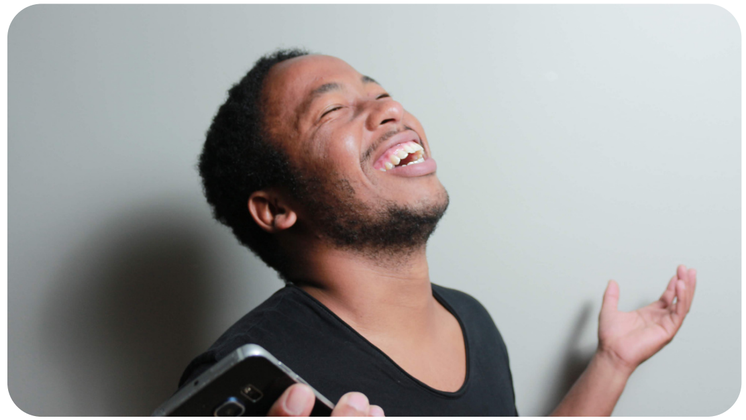
is a cashier role actually worth a human to work on it?

# Problem?

behind human cashiers

1. Waste of human labour
2. Less efficient
3. Require barcode on everything

# motivation



## AI cashier

that handles the task much more efficiently

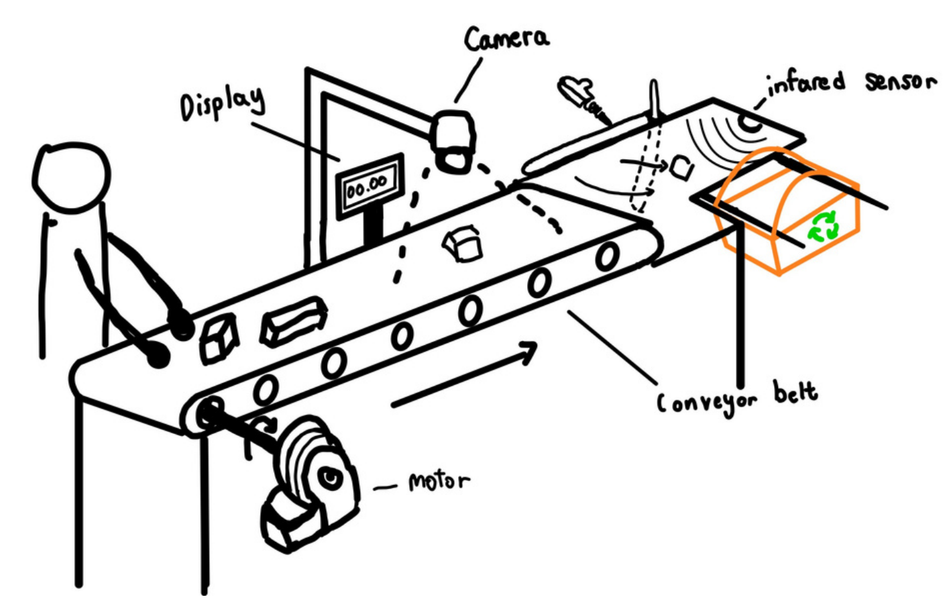
# Introducing our product

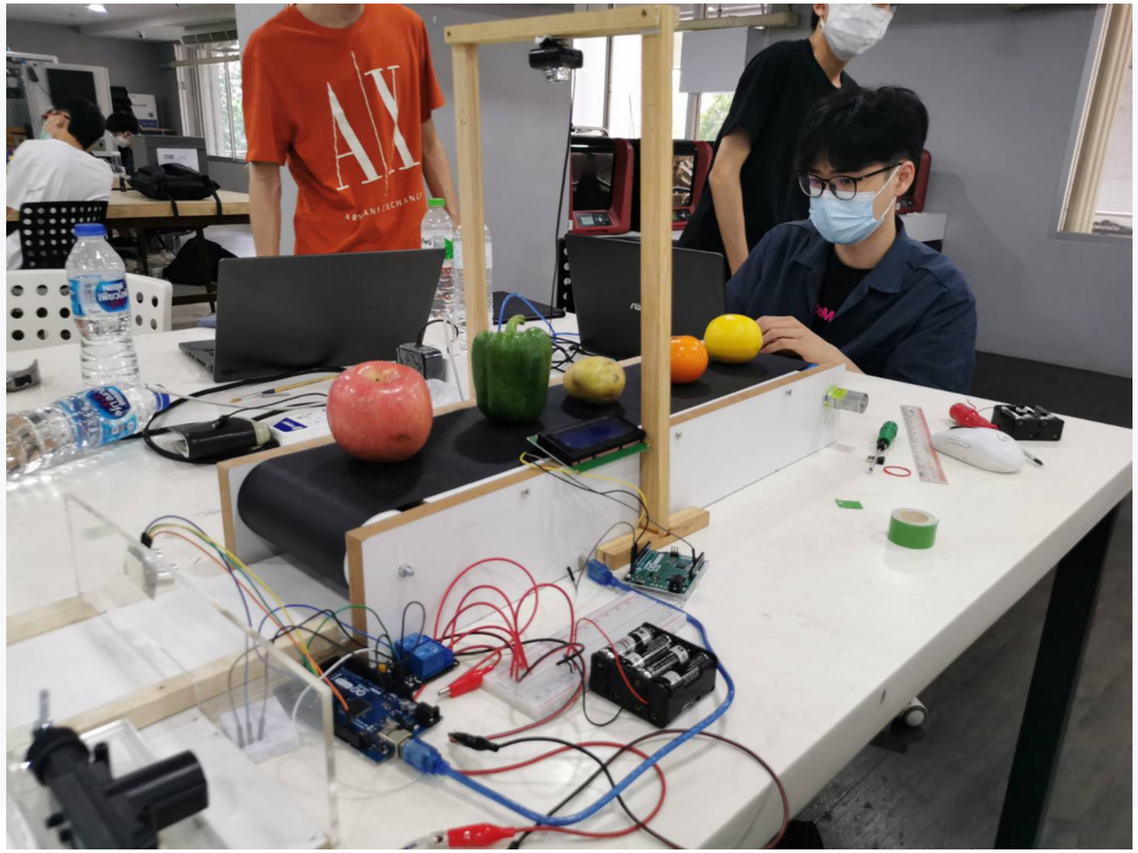
# AI cashier

# AI cashier

Before we start with the session, let's warm up a little introduction with a overall of our product

No more human cashier



# AI cashier

## How to use

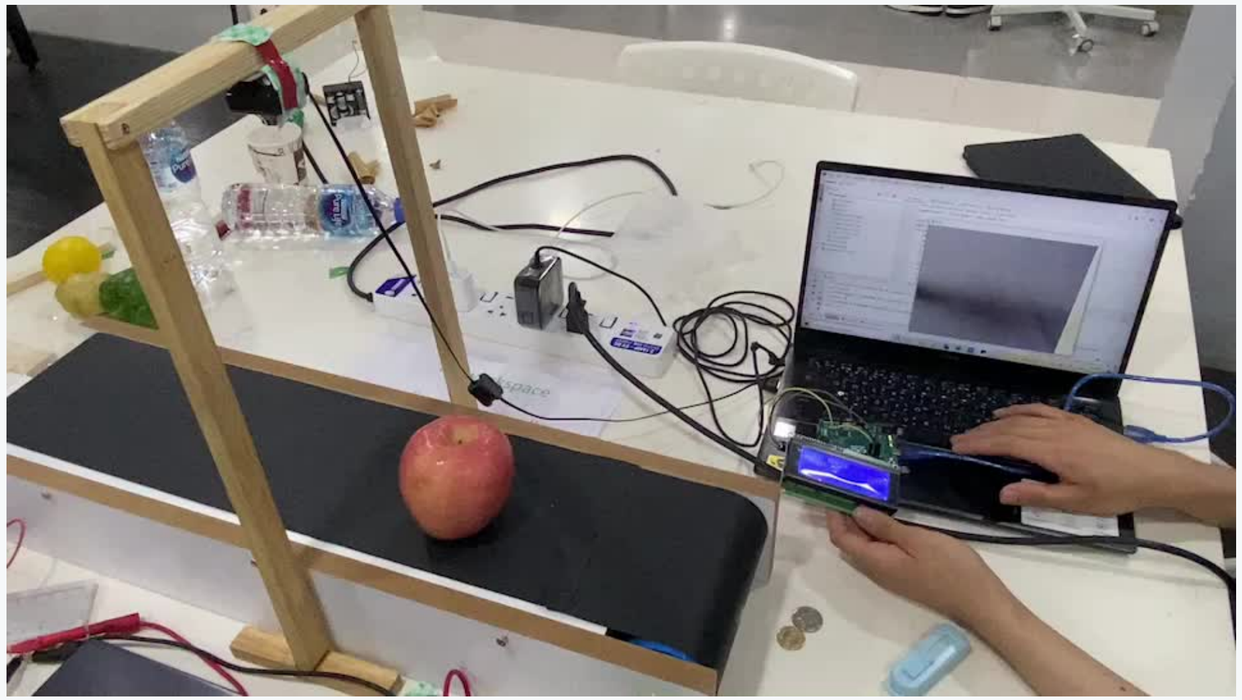
1. PUT YOUR SHOPPING ITEM ONTO THE CONVEYOR BELTS

2. THE CAMERA IN THE MIDDLE WILL PERFORM OBJECT DETECTION

3. AI WILL WILL RETRIEVE THE CORRESPONDING PRICE

4. CALCULATE THE TOTAL PRICE AND DISPLAY ON THE SCREEN

5. PASS TO THE LAST STATION WHERE TO PUSH THE OBJECT INTO THE SHOPPING BAG

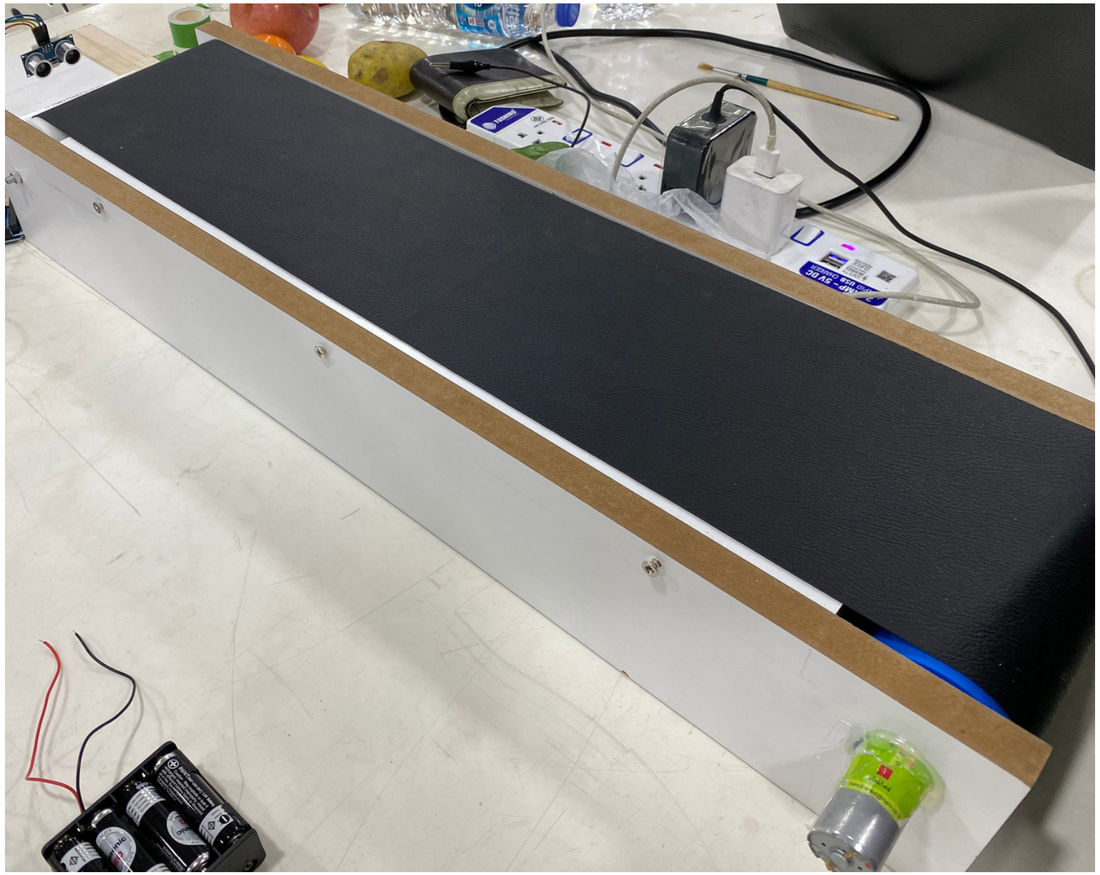


# Our componets

Explore our part in details

# CONVEYOR BELT

Use for moving the objects to the station passing the camera



length: 75 cm.
width: 20 cm.

# COMPONENTS/MATERIAL

WOOD

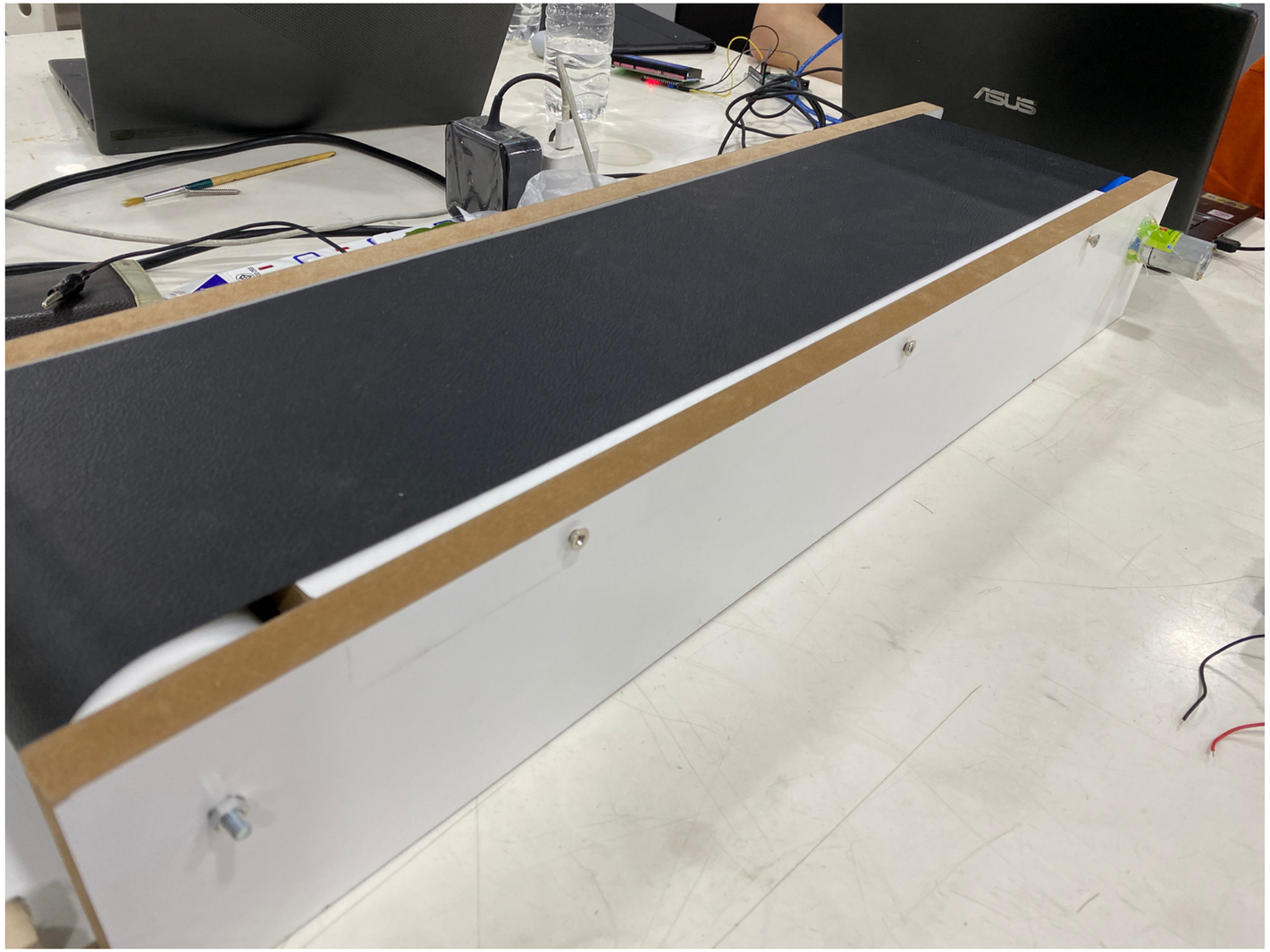
ARCYLIC (LASER CUT)

REXINE SHEET

BOLTS&NUTS

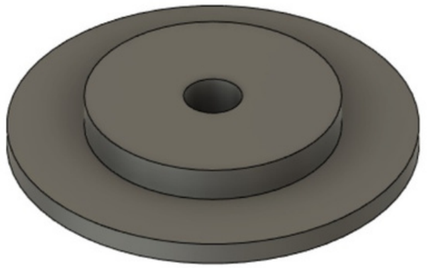
3D PRINT

MOTOR



# 3D PRINTS

HEIGHT 15 CM
DIAMETER 6.5 CM
(2X)

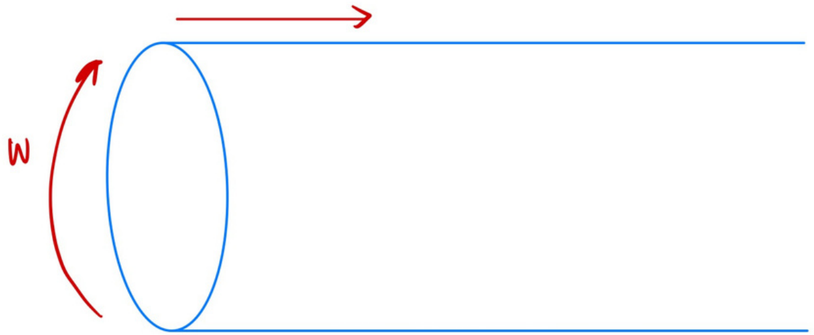


HEIGHT 2 CM
DIAMETER 4.4 CM
(4X)

# 3D PRINTS

# MOTOR

Motor selection and
motor speed calculation




มอเตอร์เกียร์ รุ่น37 แกนกลาง DC 12V:5RPM


| | |
|---|---|
| รหัสสินค้า | RM113 |
| หมวดหมู่ | Motor |
| ราคา | **380.00 บาท** |
| สถานะสินค้า | พร้อมส่ง |
| ลงสินค้า | 12 ต.ค. 2563 |
| อัพเดทล่าสุด | 20 พ.ค. 2564 |
| ความพึงพอใจ | ยังไม่มีความคิดเห็น |
| จำนวน | 1 ชิ้น |

หยิบลงตะกร้า

$$N \frac{\text{round}}{\text{min}} \times \frac{2\pi \text{ rad}}{1 \text{ round}} \times \frac{1 \text{ min}}{60 \text{ sec}} \} \text{ change to } \frac{\text{rad}}{\text{s}}$$

from $V = WR$

$$V = \left(\frac{2\pi(N) \text{ rad}}{60 \text{ s}}\right)\left(\frac{D}{2}\right)$$

$$\boxed{V = \frac{\pi N D}{60(s)}}$$

Since we want the conveyor belt move at speed of 1 - 3 $\frac{cm}{s}$ and the diameter of our 3D print is 6.5 cm.

$$(1-3)\frac{cm}{s} = \frac{\pi N (6.5)(cm)}{60(s)}$$

$$N = 2.94 \text{ rpm} - 8.81 \text{ rpm}$$

We use 5 rpm motor, So N = 5 rpm.

if N = 5 rpm then

$$V = \frac{\pi(5)(6.5)}{60} = 1.70 \frac{cm}{s}$$

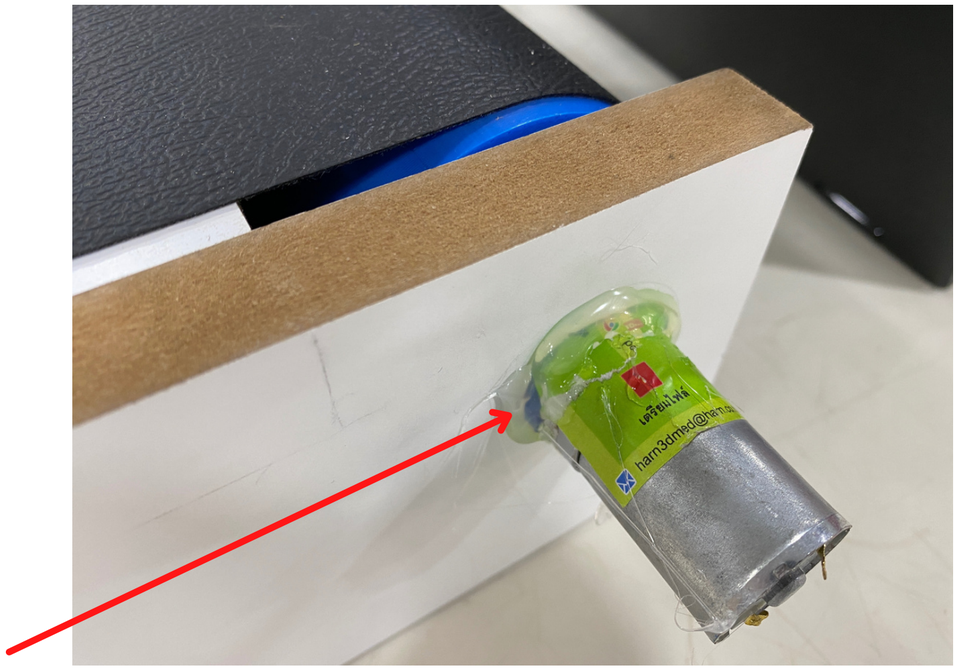
# CHALLENGES

**# Finding it hard to connect the motor rotating force with conveyor belt**

There is some reaction force while rotating the conveyor belt, resulting in the movement of motor

# SOLUTION

**Make the motor be the fixed-support with the conveyor belt (using glue)**



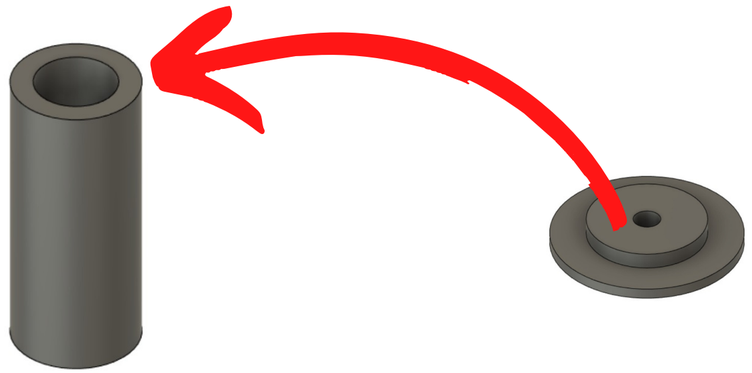
# CHALLENGES

**# The actual size of 3D print didn't match with the one we designed from program**

This make us cannot assemble assemble the smaller part with the cylinder one

# SOLUTION

**Use rasp to reduce the corner size of the 3D print**



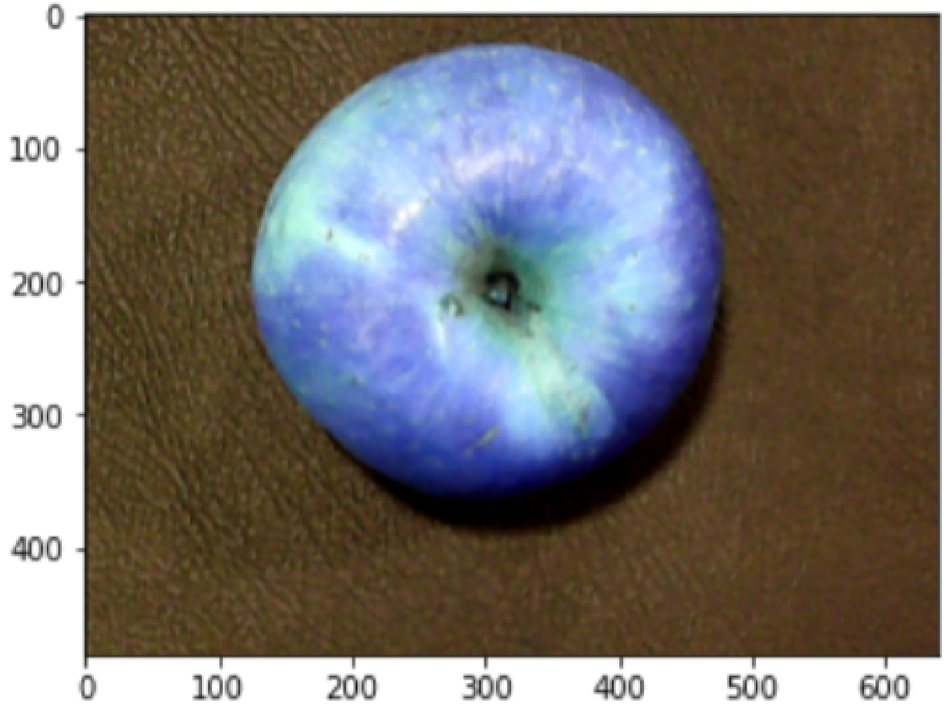
**The size of the assembling part should not be exactly equal**

# CAMERA&DISPLAY

4 PARTS

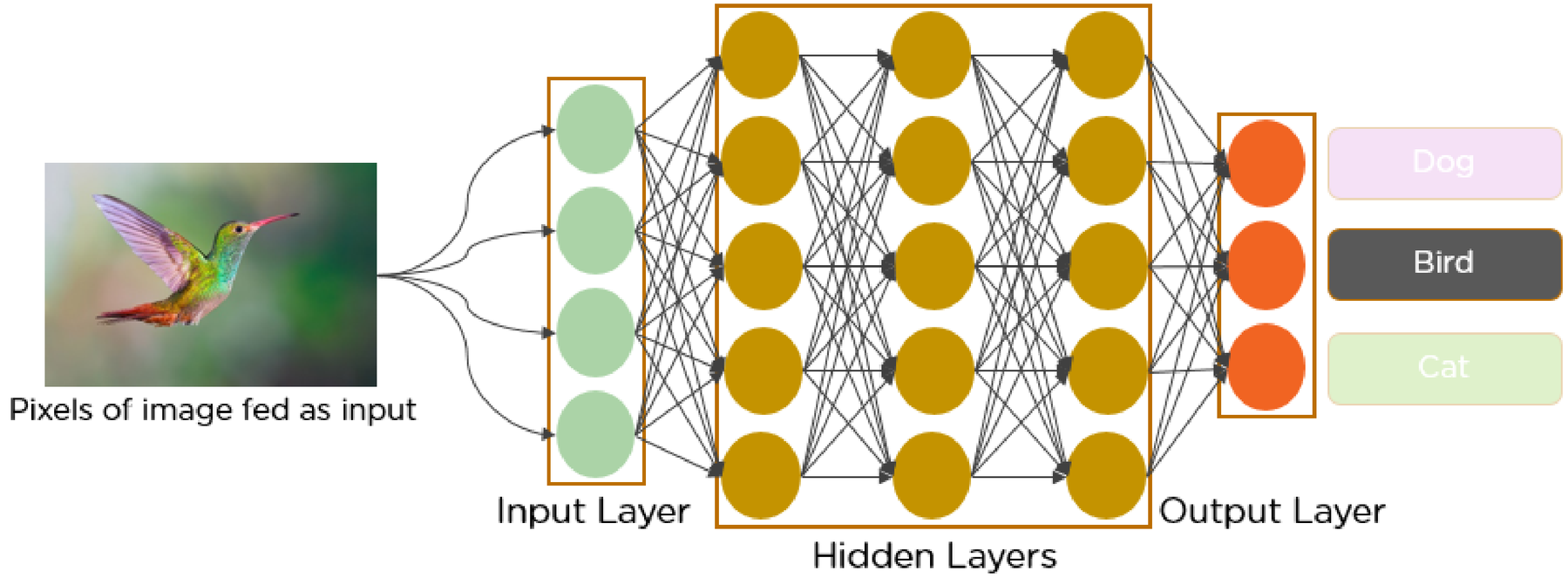
# 2.1 Camera & OpenCV

This is the second station of the AI Cashier, where we use USB 2.0 HD to capture images of the products. We use the OpenCV library for computer vision.



# 2.2 Deep Learning Model

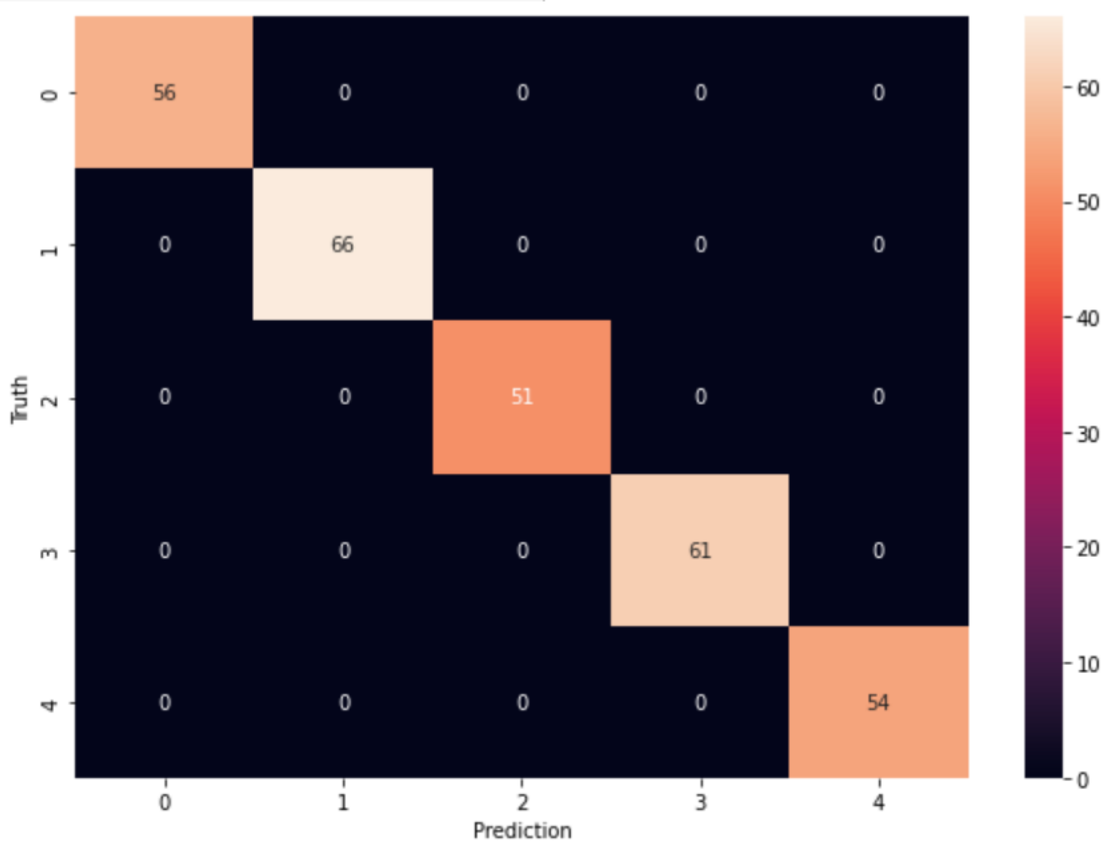
We train the model using our own dataset by taking pictures for about 50 pictures each fruit, but we use the deep learning architecture from outside.



# 2.2 Deep Learning Model

```python
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
cnn = models.Sequential([
    layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),

    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(5, activation='softmax')
])
```

```python
cnn.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```




For more details of the code:
https://github.com/gear-patt/AI_CASHIER

# 2.2 Deep Learning Model

capsicum_test1.png (unseen data)

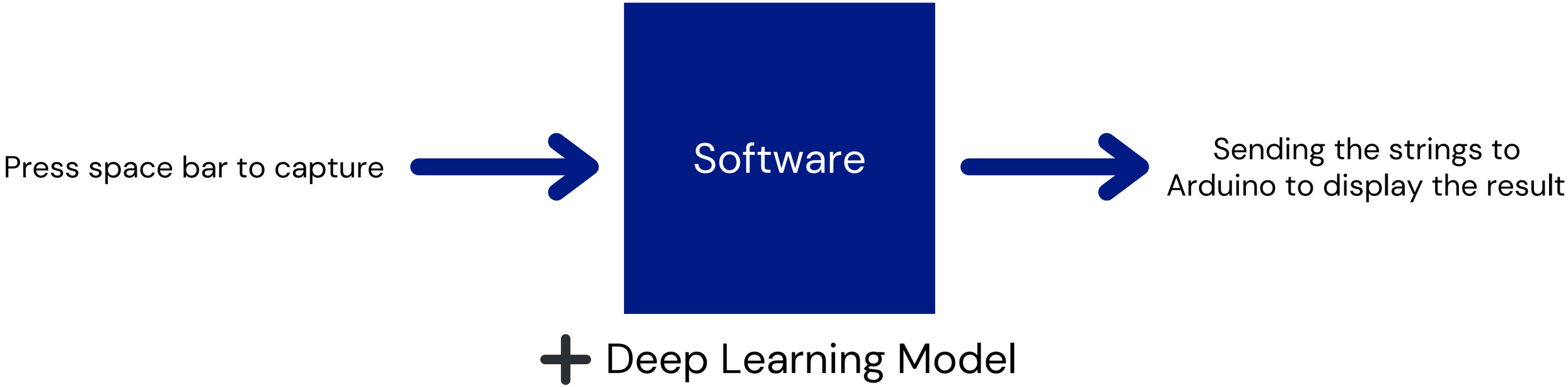
The model predicts this picture as the class3 with 99.988% probability (the highest of all 5 classes).

```
test_model('capsicum_test1.png')
```

```
[[9.7694945e-05 1.8647336e-06 9.9988079e-01 1.9586487e-05 4.4505491e-08]]
```

# 2.3 Main program: software.py

We need a program capturing images from our camera to send them to our model to predict and send the outputs, which are the prices and the types of fruits, to the Arduino.



# 2.3.1 Image capturing: data_collection.py

data_collection.py (Python)
The function of this program is to capture images for data collection. By pressing the space bar, the images of fruits are captured.

```python
import cv2

cam = cv2.VideoCapture(1)

cv2.namedWindow("test")

img_counter = 0

while True:
    ret, frame = cam.read()
    if not ret:
        print("failed to grab frame")
        break
    cv2.imshow("test", frame)

    k = cv2.waitKey(1)
    if k%256 == 27:
        # ESC pressed
        print("Escape hit, closing...")
        break
    elif k%256 == 32:
        # SPACE pressed
        img_name = "opencv_frame_{}.png".format(img_counter)
        cv2.imwrite(img_name, frame)
        print("{} written!".format(img_name))
        img_counter += 1

cam.release()

cv2.destroyAllWindows()
```

For more details of the code:
https://github.com/gear-patt/AI_CASHIER

# 2.3.2 Communication with Arduino

software.py (Python)

The program is the main program of this station, which combines the model and functions to manipulate the inputs and outputs, for instance changing from the output of the model into a single string . This program sends the output as a string, e.g. "Apple 25", to the Arduino using the serial communication. When the Arduino receives the string, the Arduino now can display the string on the LCD display.

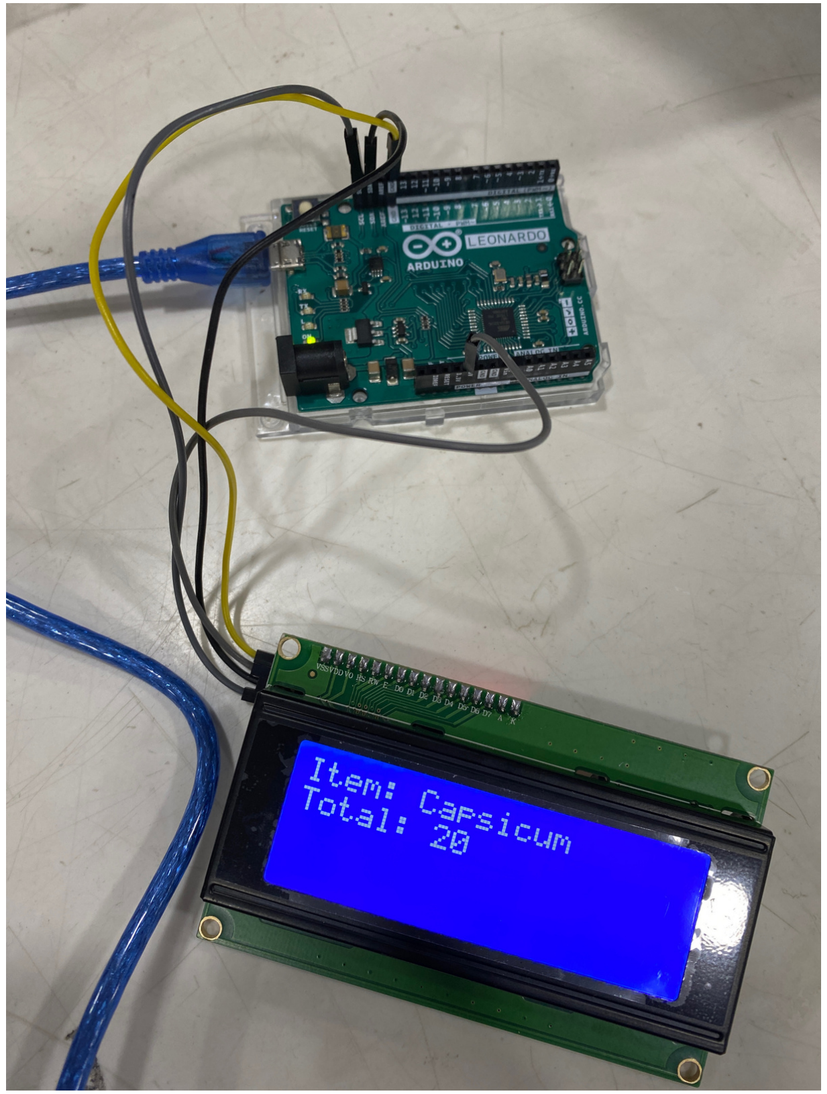
For more details of the code:
https://github.com/gear-patt/AI_CASHIER

```python
import serial
import time

cam = cv2.VideoCapture(1)

cv2.namedWindow("test")
img_counter = 0

while True:
    ret, frame = cam.read()
    if not ret:
        print("failed to grab frame")
        break
    cv2.imshow("test", frame)

    k = cv2.waitKey(1)
    text = ""
    if k % 256 == 27:
        # ESC pressed
        # apple is 25 baht, banana is 20 baht, capsicum is 15 baht, corn is 10 baht, ornage is 5 baht.
        text = "Done"
        break
    elif k % 256 == 32:
        # SPACE pressed
        img_name = "opencv_frame_{}.png".format(img_counter)
        cv2.imwrite(img_name, frame)
        img_counter += 1
        result = test_model(img_name)
        if result == 'prediction: apple':
            text = "Apple 25"
        elif result == 'prediction: banana':
            text = "Banana 20"
        elif result == 'prediction: capsicum':
            text = "Capsicum 15"
        elif result == 'prediction: corn':
            text = "Corn 10"
        else:
            text = "Orange 5"
    serialcomm = serial.Serial('/dev/cu.usbmodem14401', 9600)
    serialcomm.timeout = 1
    if text == 'Done':
        break
    if text != '':
        print(text)
        serialcomm.write(text.encode())
        time.sleep(0.01)

serialcomm.close()

cam.release()

cv2.destroyAllWindows()
```

# 2.4 Display

We use an Arduino to display the output on the 20X4 LCD display with I2C adapter.

```
void loop() {
  if(Serial.available()>0) {
    label = Serial.readStringUntil(' ');
    if(label != ""){
      price = Serial.readStringUntil('\n');
      if(price == "reset") {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Item: -");
        lcd.setCursor(0, 1);
        lcd.print("Total: 0"); // ending message can be changed
        totalPrice = 0;
      }
      else {
        totalPrice += price.toInt();
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Item: " + label);
        lcd.setCursor(0, 1);
        lcd.print("Total: " + String(totalPrice));
      }
    }

  }
}
```



# CHALLENGES

**# Communication of Python program and Arduino**

# SOLUTION

**Serial Communication is used for communicating between Python program and the Arduino.**

# Flipper Station

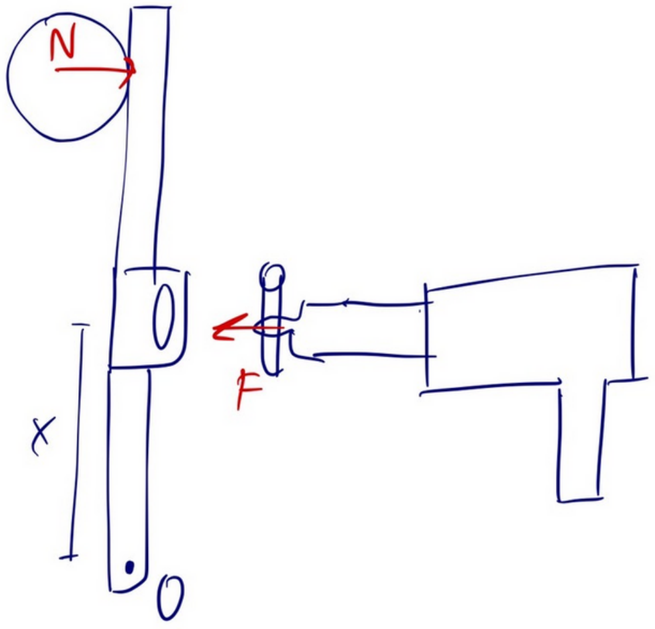
4 PARTS

# 3.1 Physical Assembly

The door linear actuator pushes and pulls the wooden stick via a slot made from laser-printed acryllic.

$$\Sigma M_0 = 0; \quad xF - lN = 0$$

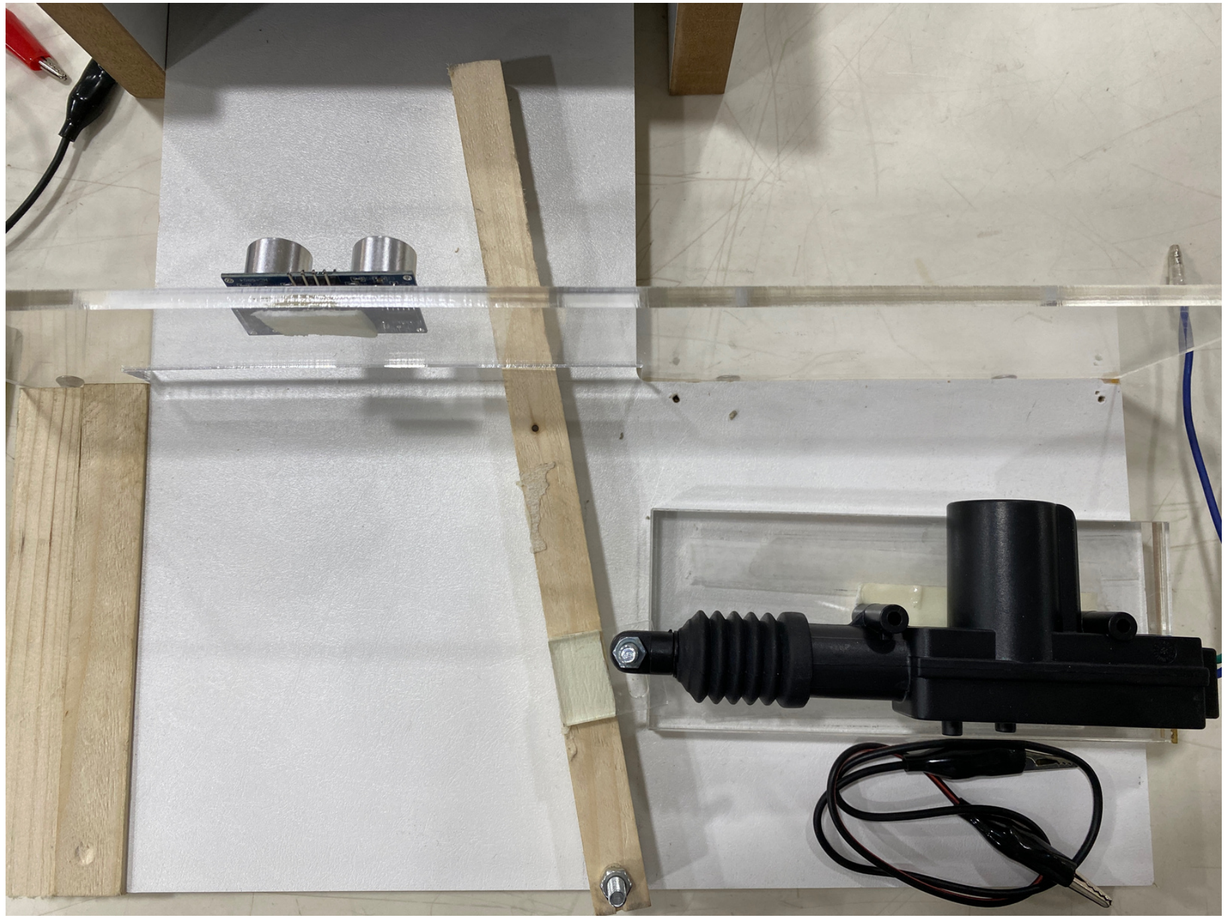
$$F = \frac{lN}{x}$$




Without the slot the stick cannot move while other components are stationary.

# 3.1 Physical Assembly

The door linear actuator pushes and pulls the wooden stick via a slot made from laser-printed acryllic.

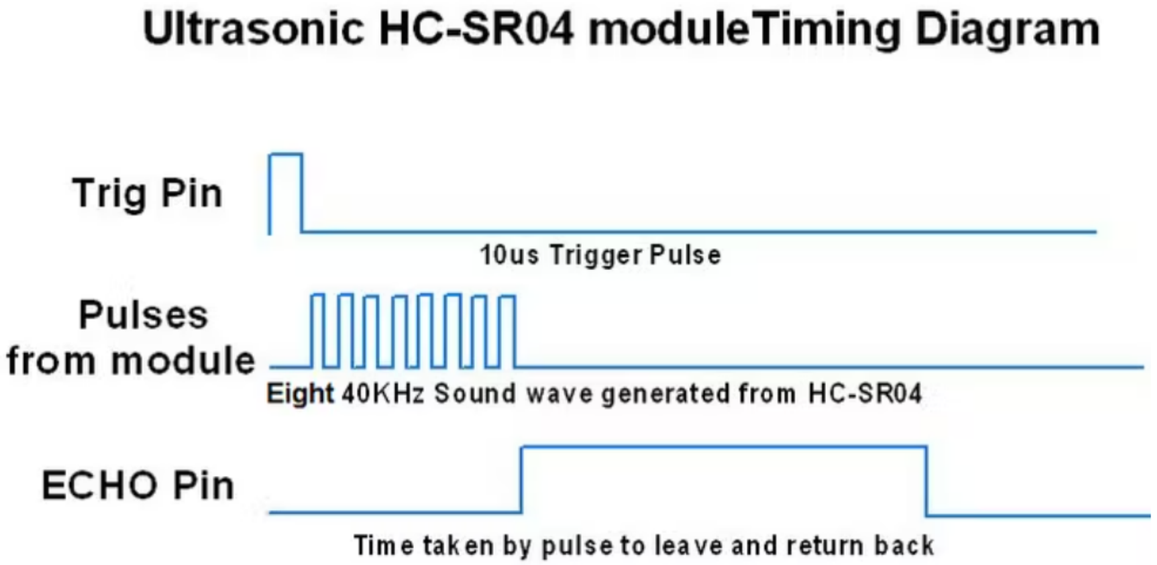


# 3.2 Ultrasonic distance sensor

The sensor pings an ultrasonic pulse then measures the time before the back-echo is detected.

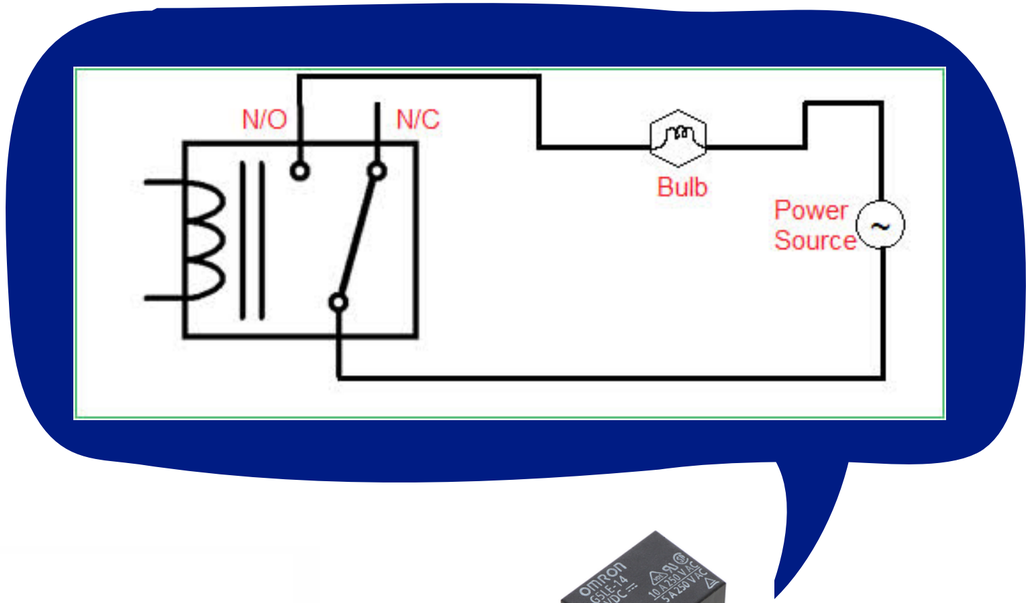
```
long duration, distance;

pinMode(pingPin, OUTPUT);

digitalWrite(pingPin, LOW);
delayMicroseconds(2);
digitalWrite(pingPin, HIGH);
delayMicroseconds(5);
digitalWrite(pingPin, LOW);
pinMode(inPin, INPUT);
duration = pulseIn(inPin, HIGH);

distance = microsecondsToCentimeters(duration);
```

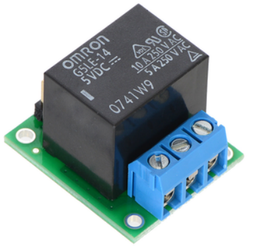
```
long microsecondsToCentimeters(long microseconds)
{
  return microseconds / 29 / 2;
}
```

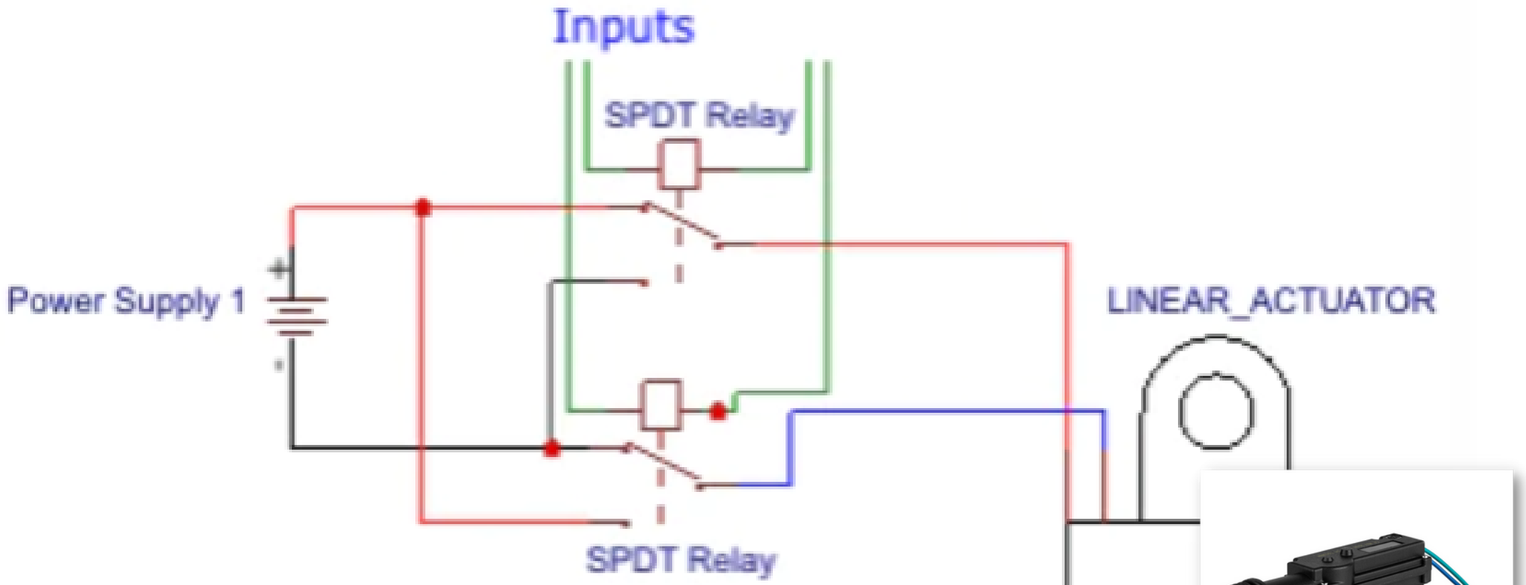



# 3.3 Controlling the actuator with relays

Two relays are used to control the 12V current that activates the flipper.




Retract when
**HIGH-LOW**
Extend when
**LOW-HIGH**
Inactive when
**HIGH-HIGH**

# 3.3 Controlling the actuator with relays

Two relays are used to control the 12V current that activates the flipper.

```
if(distance <=3.5) {
  delay(350);
  digitalWrite(4, HIGH);
  digitalWrite(8, LOW);
  delay(600);
  digitalWrite(4, LOW);
  digitalWrite(8, HIGH);
  delay(600);
}
else {
  digitalWrite(4, HIGH);
  digitalWrite(8, HIGH);
}
```

# 3.4 Challenges

1. The IR distance sensor's low-quality wires make the wiring very flimsy.

   **>>> Switch to an ultrasonic distance sensor.**

2. The item tends to roll away from the flipping zone.

   **>>> Attach an acryllic pane to stop the item from rolling.**

3. There is seemingly not enough force to push heavy items.

   **>>> Strengthen the connection. Make the slot smaller.**

4. The wire melts from high voltage/current.

   **>>> Use bigger wires.**

# lesson learnt

# Our journey

**Milestone Week#1**

Finalize hardware selection and calculations

**Milestone Week#2**

Assembling of different parts of the projects and wiring with Arduino control

**Milestone Week#3**

Connect all the parts as a whole and start to experiment and demo

**Ultimate Goal #Week 4**

Prototype testing and modification for future improvement

# further improvement

"We believe our product is a new revolution to the digital cashier world"

AI cashier

# Thank you!

Have a great day ahead.