PC Based Vending Machine

Although the author and publisher have made every effort to ensure that the information in this writing was correct at press time, the author and publisher do not assume and hereby disclaim any liability to any party for any loss, damage, or disruption caused by errors or omissions, whether such errors or omissions result from negligence, accident, or any other cause.

paid link

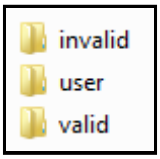
The main purpose of vending machines is to sell products to consumers without a cashier. In this writing, we will build a simple, PC based vending machine using Arduino and Python, a general purpose programming language. For the purpose of testing, we do not need anything fancy. A cheap webcam is used for this testing.

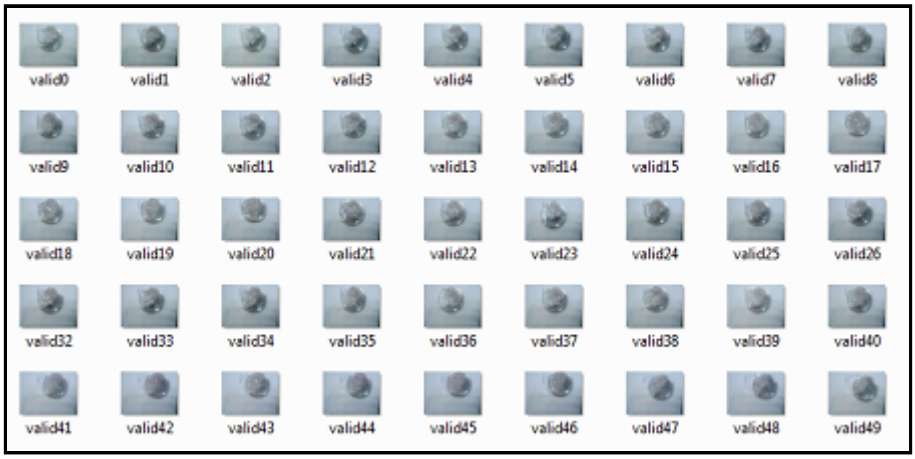
List of components:
-Personal Computer (1)
-Arduinu Uno (1)
-Web Camera (1)
-Ultrasonic Sensor (1)
-Stepper Motor NEMA23 (1)
-Stepper Motor Driver ST-M5045 (1)
-AC to DC Converter (100/220VAC to 24VDC) (1)
-Wire

How it works:
-Coin is detected by the ultrasonic sensor.
-Arduino tells Python to activates webcam.
-Webcam captures coin image.
-Using CNN, the image is compared with a bunch of images (training data) to determine whether it is a valid

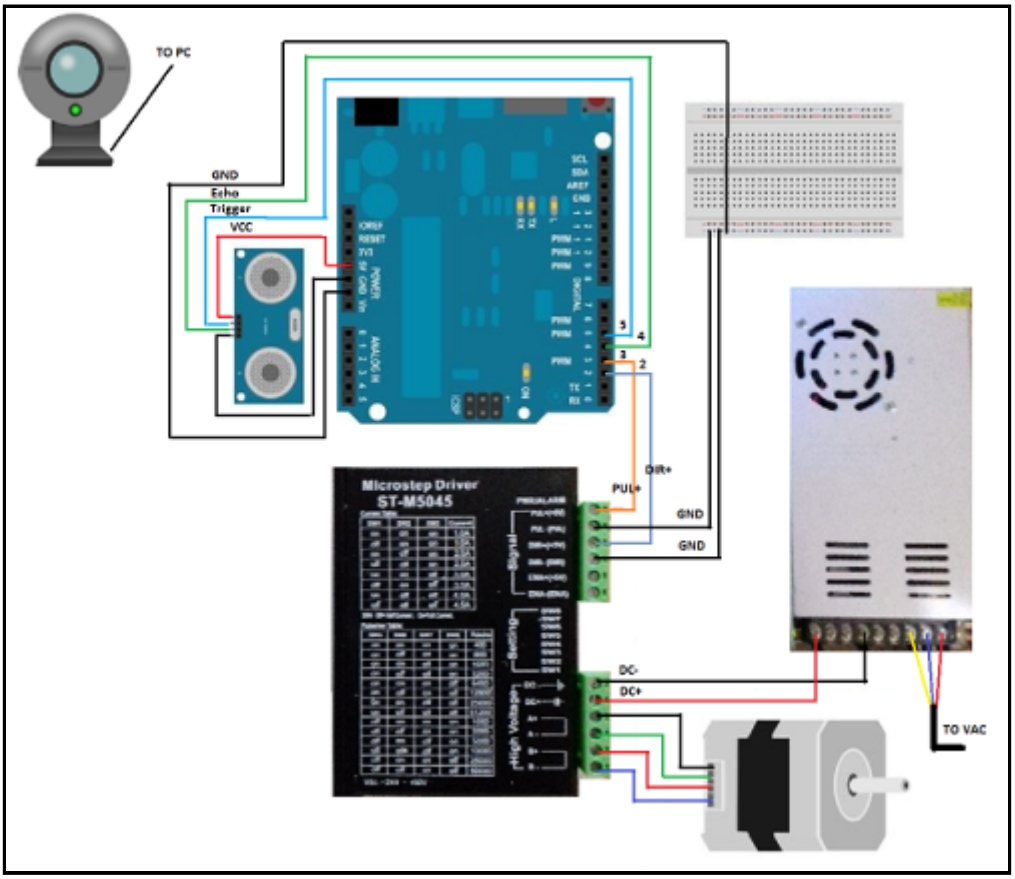
coin.
-If it is valid, Python tells Arduino to activates stepper motor.
-The purchased item is dispensed using a spiral turn mechanism. (Not yet completed)

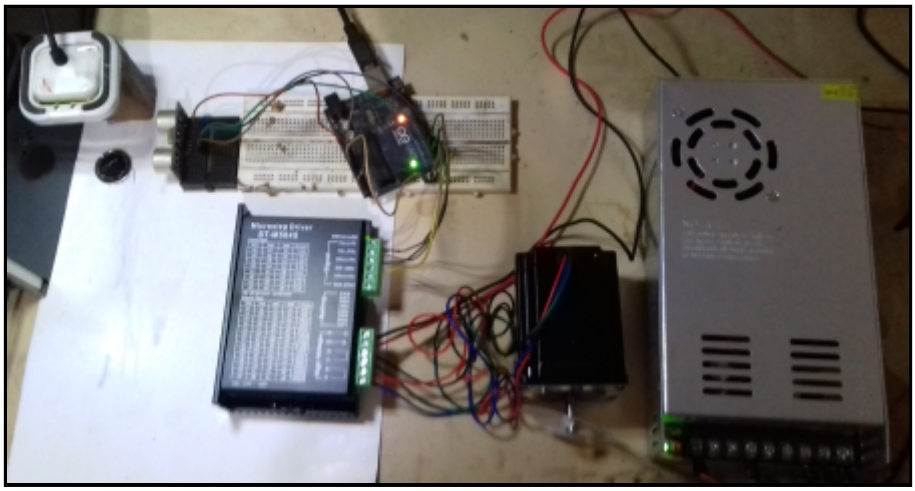
To start with, capture images for training data. Get valid images of the coin and save the images to a directory named 'valid'. Do the same for wrong coins. This can be any coin other than the valid one. This testing uses 50 images for each category. Also create directory named 'user' to save the captured image of the submitted (read: inserted) coin.





Here is the circuit:





Notes: -The webcam is facing downward. -The ultrasonic sensor can be replaced with an infrared. -If you use a dual voltage AC to DC converter, do not forget to select the right voltage. -Set the switches on the stepper motor driver appropriately. For this testing, max current of the stepper motor is 2.8A. Microstepping is set to 1600 but you can set it to 200 or any value. -Your stepper components or other components may be different than mine.

The project consists of three code files (one Arduino sketch and two Python files). Here they are:

VendingMachine.ino

```
const int echo = 4;
const int trigger = 5;
const int LED = 13;

const int driverDIR1 = 2;
const int driverPUL1 = 3;

const int microStep = 1600;

double duration;
double distance;
const double maxdistance = 5;

int pd = 500;   //pulse delay period

char inChar;
char stepperActive = '0';

void setup() {
  Serial.begin(9600);

  pinMode(trigger,OUTPUT);
  pinMode(echo,INPUT);
  pinMode(LED,OUTPUT);

  pinMode(driverDIR1,OUTPUT);
  pinMode(driverPUL1,OUTPUT);

  digitalWrite(trigger,LOW);
  digitalWrite(echo,LOW);
  digitalWrite(LED,LOW);
}

void loop() {
  digitalWrite(trigger,HIGH);
  delayMicroseconds(1000);
  digitalWrite(trigger,LOW);

  duration = pulseIn(echo,HIGH);
  distance = (duration/2)/29.1;
```

```
  if(distance < maxdistance){
    Serial.println('1');
    Serial.write(255);
    digitalWrite(LED,HIGH);
    delay(10000);
    serialEvent();
    if(stepperActive == '1') {
      for(int i=0;i<microStep;i++){
        digitalWrite(driverDIR1, LOW);
        digitalWrite(driverPUL1, HIGH);
        delayMicroseconds(pd);
        digitalWrite(driverPUL1, LOW);
        delayMicroseconds(pd);
      }
      stepperActive = '0';
    }
  }else{
    digitalWrite(LED,LOW);
  }

  delay(100);
}

void serialEvent() {
  if (Serial.available()) {
    inChar = (char)Serial.read();

    if (inChar == '1'){
        stepperActive = '1';
    }else{
        stepperActive = '0';
    }
  }
}
```

Explanation

The Arduino sketch uses serial connection to communicate with the Python program, *VendingMachine.py*.

```
if(distance < maxdistance){
    Serial.println('1');
    Serial.write(255);
   ...
}
```

When a coin is detected by the ultrasonic sensor, a message is sent to the program.

```
void serialEvent() {
  if (Serial.available()) {
    inChar = (char)Serial.read();

    if (inChar == '1'){
        stepperActive = '1';
    }else{
        stepperActive = '0';
    }
  }
}
```

The purpose of the above function is to receive the reply from the Python program. *inChar == '1'* means coin is valid.

```
if(stepperActive == '1') {
      for(int i=0;i<1600;i++){
        digitalWrite(driverDIR1, LOW);   //rotate left (viewed from the back of the stepper)
        //digitalWrite(driverDIR1, HIGH);    //rotate right (viewed from the back of the stepper)
        digitalWrite(driverPUL1, HIGH);
        delayMicroseconds(pd);
        digitalWrite(driverPUL1, LOW);
```

```
        delayMicroseconds(pd);
      }
      stepperActive = '0';
    }
```

If the coin is valid, the stepper motor rotates.

MakeModel.py

```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.layers.normalization import BatchNormalization
from PIL import Image
import numpy as np
import os

img_dir = 'images/'
img_data = []
lbl_data = []
img_size = 256

def get_file_list(directory):
    for dir_path, dir_names, file_names in os.walk(directory):
        for file_name in file_names:
            yield open(os.path.join(dir_path,file_name),'r',encoding='utf-8')

f_list = get_file_list(img_dir)

for f_path in f_list:
    if 'user' not in f_path.name:
        img = Image.open(f_path.name)
        img = img.convert('L')
        img = img.resize((img_size, img_size), Image.ANTIALIAS)
        img_data.append(np.array(img))
        if 'invalid' in f_path.name:
            lbl_data.append(np.array([0, 1]))
        else:
            lbl_data.append(np.array([1, 0]))

img_data = np.array(img_data).reshape(len(img_data),img_size,img_size,1)
lbl_data = np.array(lbl_data)

model = Sequential()
model.add(Conv2D(32, kernel_size=3, activation='relu', input_shape=(img_size,img_size,1)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Conv2D(64, kernel_size=3, activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Conv2D(64, kernel_size=3, activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(2, activation='softmax'))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(img_data, lbl_data, batch_size=25, epochs=10, verbose=1)
model.save('model.h5')
```

Explanation

Basically, it is an image classification problem. A CNN model is used to predict the coin validity. The file *MakeModel.py* creates the model. It will be used by another python file, *VendingMachine.py*, to determine whether the submitted is valid or invalid. CNN is a model architecture for deep learning. Deep learning is

out of scope of this short writing. Many great tutorials are available on the internet.

What the program does is it first convert the pixels of the images in the 'valid' and 'invalid' directories into a list of arrays.

```
 [array([[195, 195, 196, ..., 217, 217, 216],
[195, 195, 197, ..., 217, 216, 215],
[196, 197, 198, ..., 216, 216, 216],
...,
[195, 196, 197, ..., 214, 214, 213],
[195, 195, 196, ..., 214, 214, 214],
[194, 195, 195, ..., 214, 213, 214]], dtype=uint8), array([1, 0])]
```

The arrays are then converted into vectors using the following code:

```
img_data = np.array(img_data).reshape(len(img_data),img_size,img_size,1)
```

```
[[207]
 [207]
 [206]
 ...
 [220]
 [219]
 [220]]

[[206]
 [206]
 [207]
 ...
 [219]
 [219]
 [219]]]
```

Each image is also labeled (valid = [1,0]; invalid = [0,1]). The labeling process is conducted by viewing the location of the particular image (in valid or invalid directory?). These are then turned into vectors as expected by the CNN model.

```
lbl_data = np.array(lbl_data)
```

The following code creates the CNN model based on the vectors:

```
model = Sequential()
model.add(Conv2D(32, kernel_size=3, activation='relu', input_shape=(img_size,img_size,1)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Conv2D(64, kernel_size=3, activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Conv2D(64, kernel_size=3, activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(2, activation='softmax'))
```

Unlike a typical neural network, CNN filters parts of images. As can be seen, the CNN model above uses Sequential model type, one layer by one layer is added to the model. Multiple layers (Conv2D, MaxPooling2D, et) extract important features of the images. The parameters in the last Dense layer means that this is a binary classification problem.

```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(img_data, lbl_data, batch_size=25, epochs=10, verbose=1)
model.save('model.h5')
```

The training process begins. The created model is then saved as a file.

VendingMachine.py

```
from keras.models import load_model
from PIL import Image
from time import sleep
import numpy as np
import serial
import cv2

console = serial.Serial('COM4',9600)

counter = 1

img_size = 256
model = load_model('model.h5')
print('Starting')

if console.isOpen():
    while True:
        data = console.readline()
        print(data)
        if data == b'ÿ1 ':
            #sleep(1)
            print('Capture Image '+str(counter))
            cam = cv2.VideoCapture(0)
            img_name = 'images/user/input.png'
            ret, frame = cam.read()
            cv2.imwrite(img_name, frame)
            img = Image.open('images/user/input.png')
            img = img.convert('L')
            img = img.resize((img_size, img_size), Image.ANTIALIAS)
            result = model.predict(np.array(img).reshape(-1, img_size, img_size, 1))
            print(result)
            if result[0][0] - result[0][1] >= 0.5:      #probability must be greater than 75%
                print('valid')
                console.write(b'1')        #tell arduino to move the stepper
            else:
                print('invalid')
                #servo - coin out
            cam.release()
            counter += 1
        sleep(1)

console.close()
```

Explanation

```
console = serial.Serial('COM4',9600)
```

The above code opens serial connection.

```
model = load_model('model.h5')
```

The CNN model that we have created before is loaded.

```
            print('Capture Image '+str(counter))
            cam = cv2.VideoCapture(0)
            img_name = 'images/user/input.png'
            ret, frame = cam.read()
            cv2.imwrite(img_name, frame)
```

If there is a message (in this case a coin is detected by the ultrasonic sensor), the webcam captures the image of the coin.

```
result = model.predict(np.array(img).reshape(-1, img_size, img_size, 1))
```

The image, after turned into a vector, is submiited to the model. The model predicts the classification of the image, valid or invalid.

```
if result[0][0] - result[0][1] >= 0.5:
                print('valid')
                console.write(b'1')        #tell arduino to move the stepper
            else:
                print('invalid')
```

Only if the probability is greater than 75%, the image (coin) is considered valid. If it is valid, the program tells arduino to move the stepper.