CURRENCY DETECTOR - IOT MODULE IMPLEMENTATION

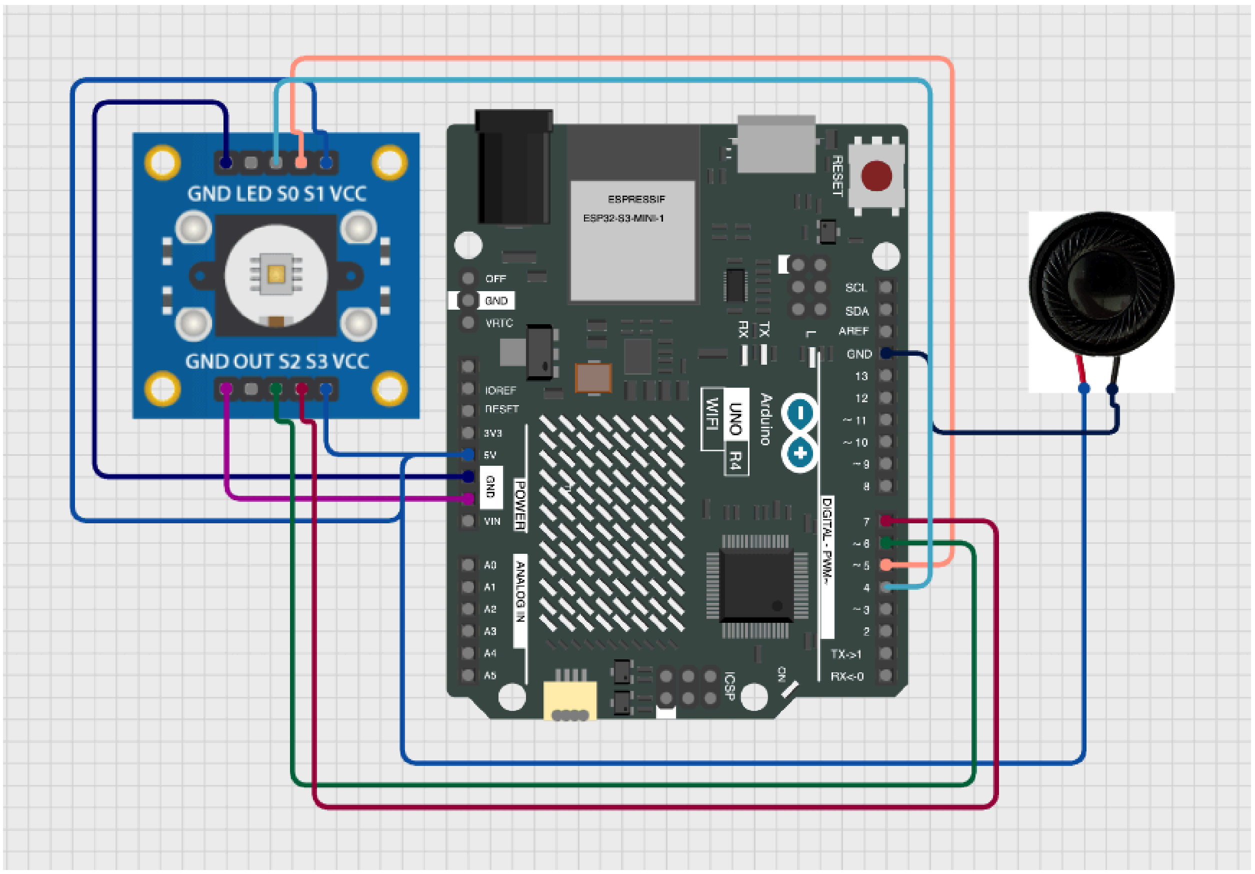
INTRODUCTION:

A currency detecting module that uses sensors, speaker and a microcontroller with the use of IOT. This is channelled for visually impaired people who find it difficult to detect the currency denomination.

COMPONENTS :

1. Arduino R4 WiFi
2. TCS3200 Color Sensor
3. Speaker
4. Jumper Wires

CIRCUIT:



PROGRAM:

ARDUINO :

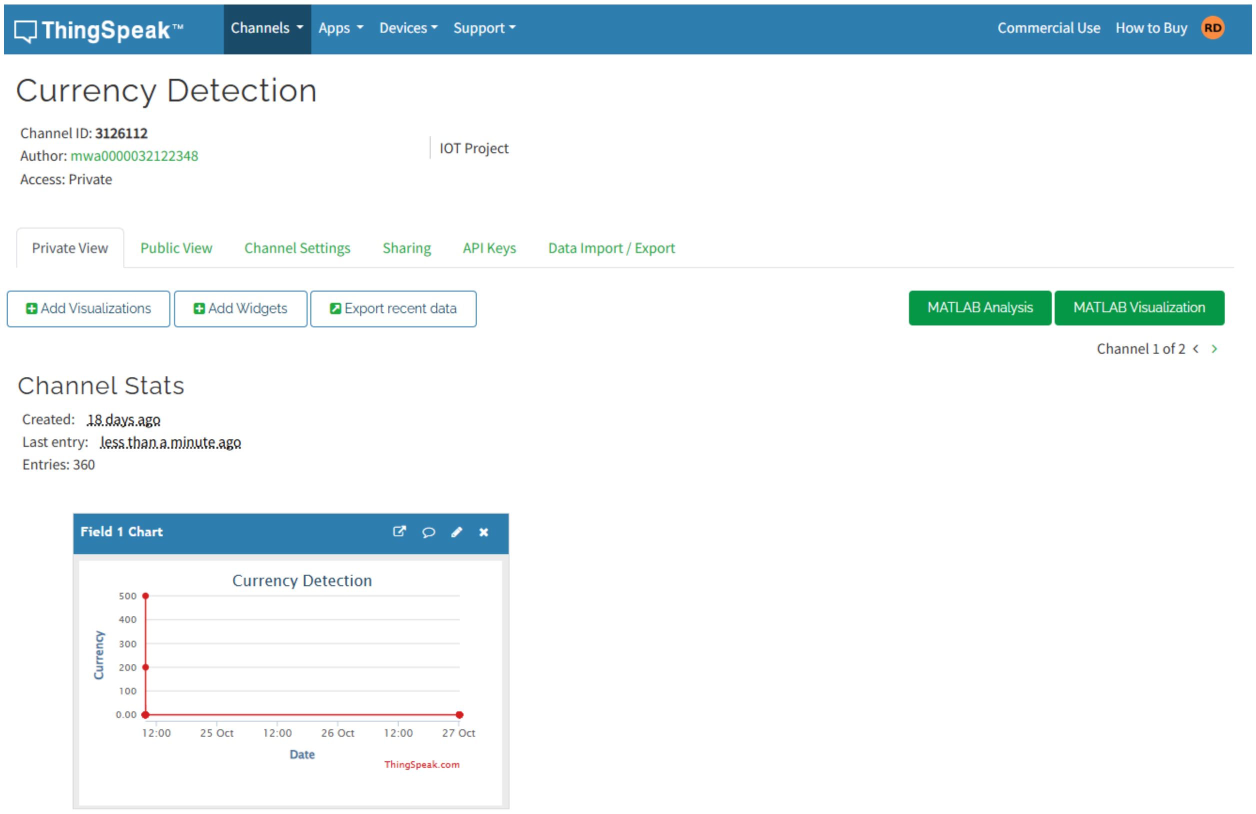
```
#include <WiFi.h>
#include "ThingSpeak.h"

// ---------- Pin Definitions ----------
int s0 = 4;
int s1 = 5;
int s2 = 6;
int s3 = 7;
int outPin = 8;
int speakerPin = 9;

// ---------- Wi-Fi & ThingSpeak ----------
const char* ssid = "Remi";
const char* password = "qwerty123";

unsigned long myChannelNumber = 3126112;
const char* myWriteAPIKey = "977JEZP3WBMGP4ZX";

WiFiClient client;

// ---------- RGB Storage ----------
int blankR, blankG, blankB;
int r10R, r10G, r10B;
int r50R, r50G, r50B;
int r100R, r100G, r100B;
int r200R, r200G, r200B;
int r500R, r500G, r500B;

bool calibrated = false;

// ---------- Setup ----------
void setup() {
  Serial.begin(115200);
  delay(1000);
  Serial.println("TCS3200 Currency Detector with ThingSpeak Ready on Arduino R4
WiFi!");

  pinMode(s0, OUTPUT);
  pinMode(s1, OUTPUT);
  pinMode(s2, OUTPUT);
  pinMode(s3, OUTPUT);
  pinMode(outPin, INPUT);
  pinMode(speakerPin, OUTPUT);

  digitalWrite(s0, HIGH);
  digitalWrite(s1, LOW);
  connectWiFi();
  ThingSpeak.begin(client);
  calibrateNotes();
}

// ---------- Loop ----------
void loop() {
  int r, g, b;
  readColor(r, g, b);

  Serial.print("R: "); Serial.print(r);
  Serial.print(" G: "); Serial.print(g);
  Serial.print(" B: "); Serial.println(b);

  String note = detectCurrency(r, g, b);
  Serial.print("  --> Detected: ");
  Serial.println(note);

  playToneForCurrency(note);
  sendToThingSpeak(note);

    delay(3000);
}

// ---------- Wi-Fi ----------
void connectWiFi() {
  Serial.println("Connecting to Wi-Fi...");
  WiFi.begin(ssid, password);

  int retry = 0;
  while (WiFi.status() != WL_CONNECTED && retry < 20) {
    delay(500);
    Serial.print(".");
    retry++;
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\n Wi-Fi Connected!");
    Serial.print("IP Address: ");
    Serial.println(WiFi.localIP());
  } else {
    Serial.println("\n Failed to connect to Wi-Fi.");
  }
}

// ---------- Color Reading ----------
void readColor(int &red, int &green, int &blue) {
  digitalWrite(s2, LOW);
  digitalWrite(s3, HIGH);
  red = pulseIn(outPin, LOW);

  digitalWrite(s2, HIGH);
  digitalWrite(s3, HIGH);
  green = pulseIn(outPin, LOW);

  digitalWrite(s2, LOW);
  digitalWrite(s3, LOW);
  blue = pulseIn(outPin, LOW);
}

// ---------- Calibration ----------
void calibrateNotes() {
  Serial.println(" Currency Calibration Mode");

  Serial.println("Show blank space...");
  delay(5000);
  readColor(blankR, blankG, blankB);
  Serial.println("Blank space calibrated!");

  Serial.println("Show ₹10 note...");
  delay(5000);
  readColor(r10R, r10G, r10B);
  Serial.println("₹10 calibrated!");

  Serial.println("Show ₹50 note...");
  delay(5000);
  readColor(r50R, r50G, r50B);
  Serial.println("₹50 calibrated!");

  Serial.println("Show ₹100 note...");
  delay(5000);
  readColor(r100R, r100G, r100B);
  Serial.println("₹100 calibrated!");

  Serial.println("Show ₹200 note...");
  delay(5000);
  readColor(r200R, r200G, r200B);
  Serial.println("₹200 calibrated!");

  Serial.println("Show ₹500 note...");
  delay(5000);
  readColor(r500R, r500G, r500B);
  Serial.println("₹500 calibrated!");

  calibrated = true;
  Serial.println(" Calibration Complete! Now detecting...");
}
```

```
// ---------- Detection ----------
String detectCurrency(int r, int g, int b) {
  if (!calibrated) return "Not Calibrated";

  long dBlank = sq(r - blankR) + sq(g - blankG) + sq(b - blankB);
  long d10 = sq(r - r10R) + sq(g - r10G) + sq(b - r10B);
  long d50 = sq(r - r50R) + sq(g - r50G) + sq(b - r50B);
  long d100 = sq(r - r100R) + sq(g - r100G) + sq(b - r100B);
  long d200 = sq(r - r200R) + sq(g - r200G) + sq(b - r200B);
  long d500 = sq(r - r500R) + sq(g - r500G) + sq(b - r500B);

  long minDist = dBlank;
  String note = "Blank Space";

  if (d10 < minDist) { minDist = d10; note = "₹10"; }
  if (d50 < minDist) { minDist = d50; note = "₹50"; }
  if (d100 < minDist) { minDist = d100; note = "₹100"; }
  if (d200 < minDist) { minDist = d200; note = "₹200"; }
  if (d500 < minDist) { minDist = d500; note = "₹500"; }

  return note;
}

// ---------- Sound Feedback ----------
void playToneForCurrency(String note) {
  int freq = 0;
  if (note == "₹10") freq = 400;
  else if (note == "₹50") freq = 600;
  else if (note == "₹100") freq = 700;
  else if (note == "₹200") freq = 500;
  else if (note == "₹500") freq = 900;
  else if (note == "Blank Space") freq = 0;

  if (freq > 0) {
    tone(speakerPin, freq, 300);
    delay(350);
  }
}

// ---------- ThingSpeak ----------
void sendToThingSpeak(String note) {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("Wi-Fi disconnected, reconnecting...");
    connectWiFi();
  }

  int value = 0;
  if (note == "₹10") value = 10;
  else if (note == "₹50") value = 50;
  else if (note == "₹100") value = 100;
  else if (note == "₹200") value = 200;
  else if (note == "₹500") value = 500;
  else if (note == "Blank Space") value = 0;

  ThingSpeak.setField(1, value);
  int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);

  if (x == 200) {
    Serial.println(" ThingSpeak Update Successful!");
  } else {
    Serial.print(" ThingSpeak Update Failed, Code: ");
    Serial.println(x);
  }
}
```

PYTHON:

```
import time
import RPi.GPIO as GPIO
import requests

# ---------- Pin Definitions ----------
s0 = 4
s1 = 5
s2 = 6
s3 = 7
outPin = 8
speakerPin = 9

# ---------- Wi-Fi & ThingSpeak ----------
THINGSPEAK_API_KEY = "977JEZP3WBMGP4ZX"
THINGSPEAK_CHANNEL = 3126112
THINGSPEAK_URL = "https://api.thingspeak.com/update"

# ---------- RGB Storage ----------
blank = {}
r10 = {}
r50 = {}
r100 = {}
r200 = {}
r500 = {}

calibrated = False

# ---------- GPIO Setup ----------
GPIO.setmode(GPIO.BCM)
GPIO.setup(s0, GPIO.OUT)
GPIO.setup(s1, GPIO.OUT)
GPIO.setup(s2, GPIO.OUT)
GPIO.setup(s3, GPIO.OUT)
GPIO.setup(outPin, GPIO.IN)
GPIO.setup(speakerPin, GPIO.OUT)

GPIO.output(s0, GPIO.HIGH)
GPIO.output(s1, GPIO.LOW)

# ---------- Helper: Pulse Measurement ----------
def read_pulse():
    start = time.time()
    while GPIO.input(outPin) == 0:
        pass
    start = time.time()
    while GPIO.input(outPin) == 1:
        pass
    end = time.time()
    return (end - start) * 1000000  # microseconds

# ---------- Color Reading ----------
def read_color():
    # RED
    GPIO.output(s2, GPIO.LOW)
    GPIO.output(s3, GPIO.HIGH)
    red = read_pulse()

    # GREEN
    GPIO.output(s2, GPIO.HIGH)
    GPIO.output(s3, GPIO.HIGH)
    green = read_pulse()

    # BLUE
    GPIO.output(s2, GPIO.LOW)
    GPIO.output(s3, GPIO.LOW)
    blue = read_pulse()

    return (int(red), int(green), int(blue))

# ---------- Calibration ----------
def calibrate_notes():
    global blank, r10, r50, r100, r200, r500, calibrated
    print(" Currency Calibration Mode")

    blank = capture_color("Blank Space")
    r10 = capture_color("₹10")
    r50 = capture_color("₹50")
    r100 = capture_color("₹100")
    r200 = capture_color("₹200")
    r500 = capture_color("₹500")

    calibrated = True
    print(" Calibration Complete! Now detecting...")

def capture_color(label):
    print(f"Show {label}...")
    time.sleep(5)
    r, g, b = read_color()
    print(f"{label} Calibrated -> R:{r}, G:{g}, B:{b}")
    return {"r": r, "g": g, "b": b}

# ---------- Detection ----------
def detect_currency(r, g, b):
    if not calibrated:
        return "Not Calibrated"

    samples = {
        "Blank Space": blank,
        "₹10": r10,
        "₹50": r50,
        "₹100": r100,
        "₹200": r200,
        "₹500": r500
    }

    min_dist = None
    detected = "Blank Space"

    for note, val in samples.items():
        dist = (r - val["r"])**2 + (g - val["g"])**2 + (b - val["b"])**2
        if min_dist is None or dist < min_dist:
            min_dist = dist
            detected = note

    return detected

# ---------- Sound Feedback ----------
def play_tone(note):
    import os
    freqs = {
        "₹10": 400,
        "₹50": 600,
        "₹100": 700,
        "₹200": 500,
        "₹500": 900,
        "Blank Space": 0
    }
    f = freqs.get(note, 0)
    if f > 0:
        os.system(f"beep -f {f} -l 300")  # Linux 'beep' command
        time.sleep(0.35)

# ---------- ThingSpeak ----------
def send_to_thingspeak(note):
    value_map = {
        "₹10": 10,
        "₹50": 50,
        "₹100": 100,
        "₹200": 200,
        "₹500": 500,
        "Blank Space": 0
    }
    value = value_map.get(note, 0)

    payload = {"api_key": THINGSPEAK_API_KEY, "field1": value}
    response = requests.post(THINGSPEAK_URL, data=payload)
    if response.status_code == 200:
        print(" ThingSpeak Update Successful!")
    else:
        print(f" ThingSpeak Update Failed: {response.status_code}")

# ---------- Main Loop ----------
if __name__ == "__main__":
    print("TCS3200 Currency Detector + ThingSpeak (Python Edition) Ready!")
    calibrate_notes()

    while True:
        r, g, b = read_color()
        print(f"R:{r} G:{g} B:{b}")
        note = detect_currency(r, g, b)
        print(f"  --> Detected: {note}")
        play_tone(note)
        send_to_thingspeak(note)
        time.sleep(3)
```

THINGSPEAK VISUALIZATION:



Currency Detection

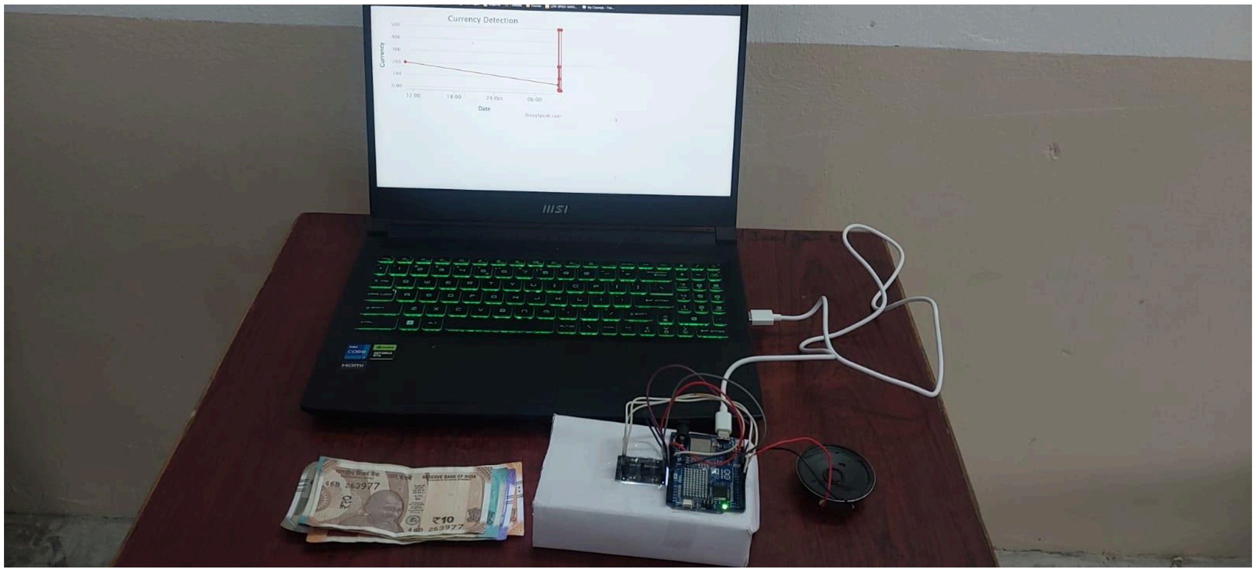
Channel ID: 3126112
Author: mwa0000032122348
Access: Private

IOT Project

Channel Stats

Created: 18 days ago
Last entry: less than a minute ago
Entries: 360

IMPLEMENTED MODULE :



TEAM MEMBERS :

1. Ram Siddhaarth - 311123106049
2. Remi Dayakar F - 311123106050
3. Jone Priyan Raj - 311123106304