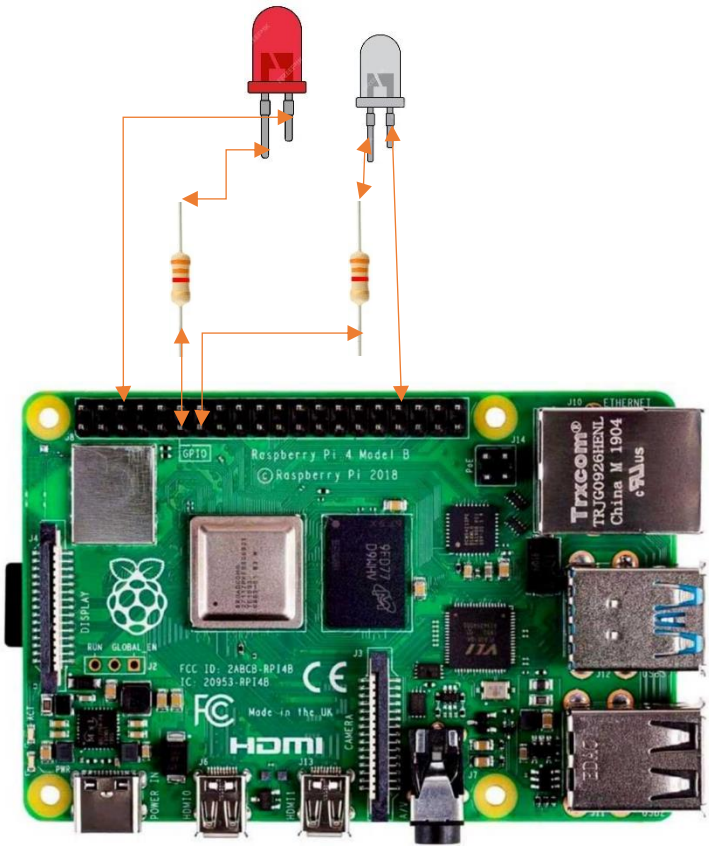
AMMETER READING DETECTOR

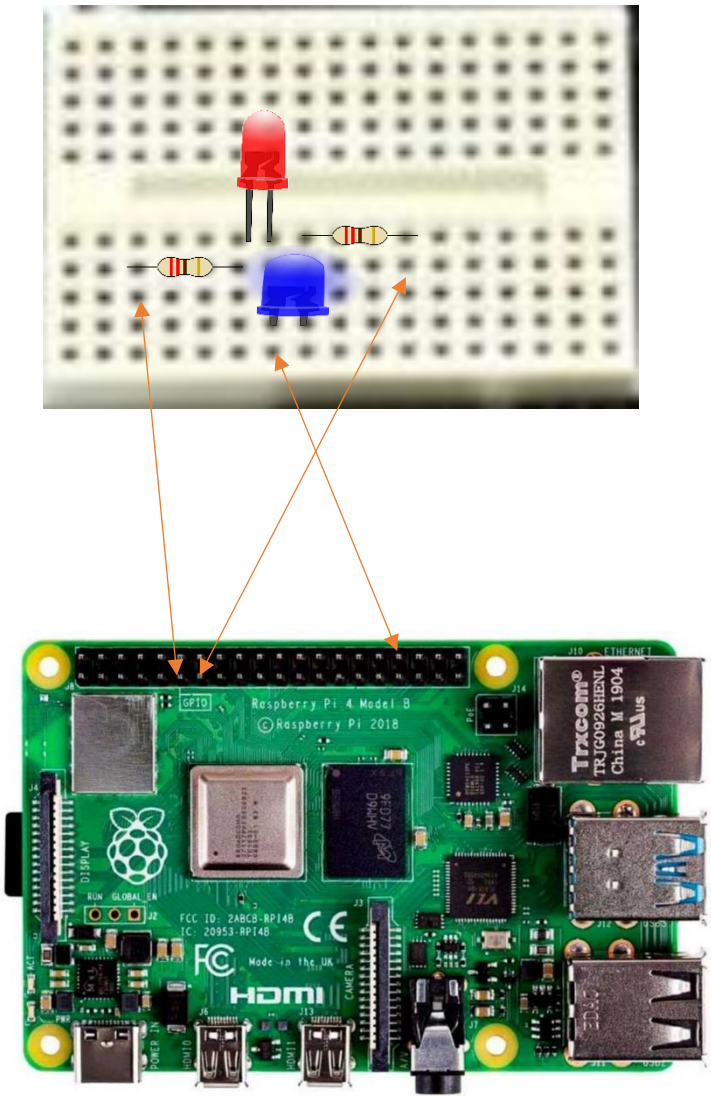
Components Used:

1. Micro Controller/Processor: Raspberry PI 4b, 4GB with 32GB SD card.
2. Power adaptor 5V/3Amps USB-C type (**need to purchase**).
3. Heat Sink for Rpi (**Optional, need to purchase**)
4. Bread board (**need to purchase**).
5. LEDs.
6. Resistors (3.3k ohms).
7. Jumper Wires.
8. Camera (for Raspberry Pi 4 B 3 B+ Camera Module Automatic IR-Cut Switching Day/Night Vision Video Module Adjustable Focus 5MP OV5647 Sensor 1080p HD Webcam for Raspberry Pi 2/3 Model B Model A A+ https://a.co/d/ikJa9uQ)
9. Mouse, keyboard and screen (**need to purchase**).

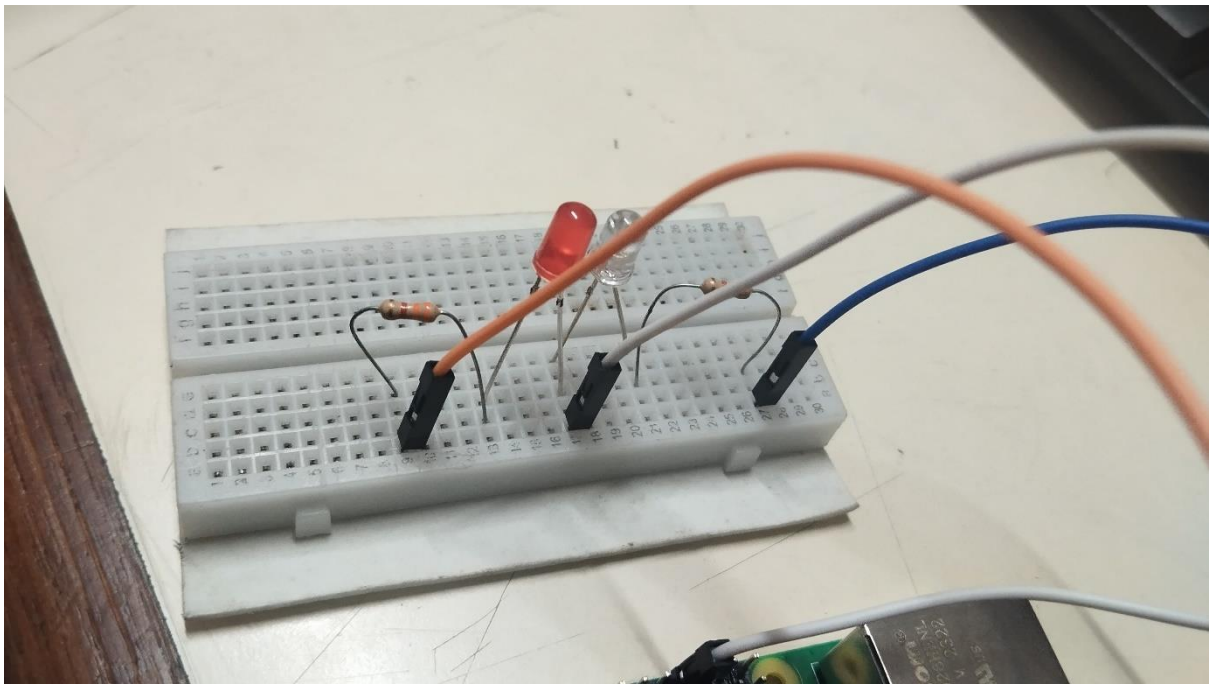
Circuit Diagram (Without Bread board):

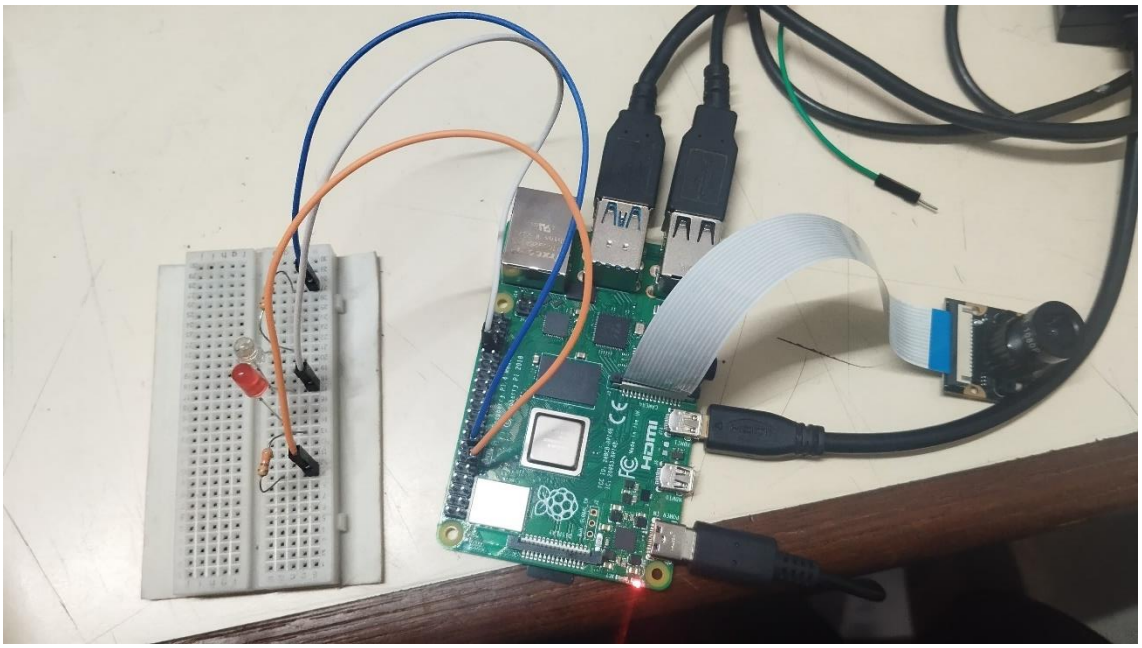


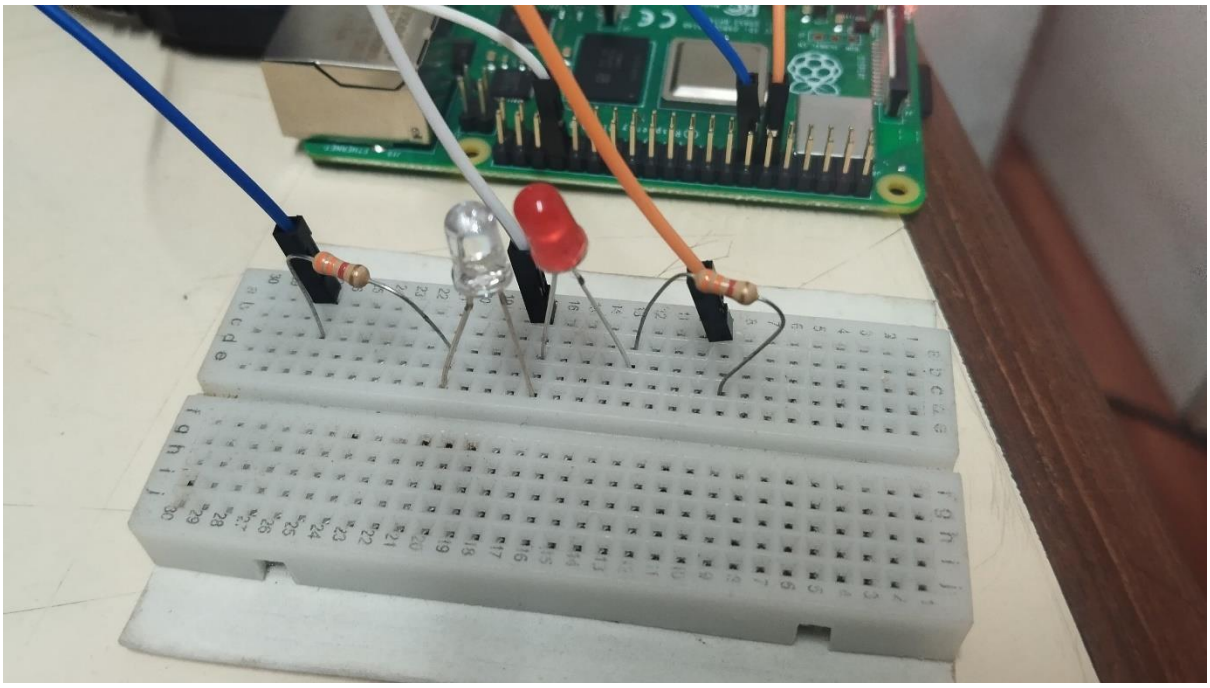
Circuit Diagram (With Breadboard):



Circuit images:







Required Packages and Installation Commands

1. OpenCV (cv2)
    - For real-time image processing and video feed handling.
    - Installation Command: **pip install opencv-python**
2. Pytesseract
    - For Optical Character Recognition (OCR) to extract text from images.
    - Installation Command: **pip install pytesseract**
    - Additional Dependency: Tesseract-OCR must be installed on the system.
        - Ubuntu: sudo apt install tesseract-ocr
        - Windows: Download and install from Tesseract GitHub.
        - Mac: brew install tesseract
3. gpiozero
    - For controlling GPIO pins of a Raspberry Pi.

- Installation Command**: pip install gpiozero**

Code Explanation

This code processes a video feed to detect numeric values in a specific region of interest (ROI) using OCR and controls GPIO LEDs to indicate missing or sequential numbers.

1. Importing Required Modules

```
import cv2
import pytesseract
from gpiozero import LED
```

- cv2: Handles camera feed and image processing.
- pytesseract: Performs OCR to extract numbers from frames.
- gpiozero.LED: Allows control of GPIO pins for LEDs.

2. Defining GPIO Pins for LEDs

```
led_upcount = LED(17)  # GPIO 17 for upcount
led_downcount = LED(27)  # GPIO 27 for downcount
```

- Two LEDs are defined: one for upcount and another for downcount.

3. Configuring Pytesseract Path

```
pytesseract.pytesseract.tesseract_cmd = r'/usr/bin/tesseract'
```

- Specifies the path to the Tesseract-OCR executable. Adjust this path based on your system.

4. Defining the detect_numbers Function

```
def detect_numbers(frame):
    roi = frame[250:550, 450:750]
```

- Captures a specific portion of the frame for OCR processing.
- The ROI is a smaller section of the frame, reducing computation time.

```
    gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
```

- Converts the ROI to grayscale for easier thresholding and better OCR accuracy.

```
    _, thresh = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
```

- Applies a binary threshold to create a high-contrast image for OCR.

```
    config = "--psm 7 --oem 3 -c tessedit_char_whitelist=0123456789"
```

- --psm 7: Treats the image as a single line of text.
- --oem 3: Enables default OCR engine.
- tessedit_char_whitelist: Restricts detection to digits only.

```
    text = pytesseract.image_to_string(thresh, config=config)
    numbers = ''.join(char for char in text if char.isdigit())
    return int(numbers) if numbers.isdigit() else None
```

- Extracts and returns integers if valid digits are detected; otherwise, returns None.

5. The main Function

```
cap = cv2.VideoCapture(0)
```

- Initializes the default camera for video capture.

```
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)
```

- Sets the camera resolution to HD (1280x720).

```
if not cap.isOpened():
    print("Error: Could not open camera.")
    return
```

- Checks if the camera opened successfully.

6. Reading and Processing Frames

```
while True:
    ret, frame = cap.read()
```

- Continuously captures frames from the camera.

```
detected_number = detect_numbers(frame)
```

- Calls detect_numbers to extract a number from the current frame.

```
if detected_number is not None and detected_number != previous_number:
```

- Ensures that the detected number is valid and different from the previous number.

7. Handling Missing Numbers with LEDs

```
if current_number == prev_number + 1:
    led_upcount.off()
    led_downcount.off()
```

- If the current number is sequential (upcount), turns off both LEDs.

```
elif current_number == prev_number - 1:
    led_upcount.off()
    led_downcount.off()
```

- If the current number is sequential (downcount), turns off both LEDs.

```
if current_number > prev_number + 1:
    led_upcount.on()
```

- If a number is missing in the upcount sequence, turns on the upcount LED.

```
elif current_number < prev_number - 1:
    led_downcount.on()
```

- If a number is missing in the downcount sequence, turns on the downcount LED.

8. Visualizing ROI and Displaying Feed

```
cv2.rectangle(frame, (450, 250), (750, 550), (0, 255, 0), 2)
```

- Draws a rectangle around the ROI for visualization.

```
cv2.imshow("Camera Feed", frame)
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```

- Displays the camera feed with ROI. Exits on pressing the 'q' key.

9. Releasing Resources

```
cap.release()
cv2.destroyAllWindows()
```

- Releases the camera and closes all OpenCV windows.

Usage Notes

1. Ensure Tesseract-OCR is correctly installed and its path is configured.
2. Connect LEDs to GPIO pins 17 and 27 of your Raspberry Pi.
3. Adjust the ROI dimensions to match your camera's field of view and the location of numbers to be detected.

Steps to Run the Code:

1. **Automatic Execution on Startup**:
   The with_decimal.py script is configured to run automatically in the background when the Raspberry Pi is powered on.

2. **Viewing Output**:
   To view the output, click on the **LXTerminal** located in the top-left corner of the desktop.

3. **Stopping Execution**:
   To stop the execution, press **'q'** on the camera screen.

4. **Editing Files**:
   If you wish to modify the code, navigate to:
   Files -> Documents
   Locate the files named with_decimal.py and without_decimal.py. Open the desired file in a code editor to make any necessary changes.

5. **Running the Code Manually**:
   To manually run either script, open a terminal and use the following command:

   python3 filename.py

   Replace filename.py with either with_decimal.py or without_decimal.py as required