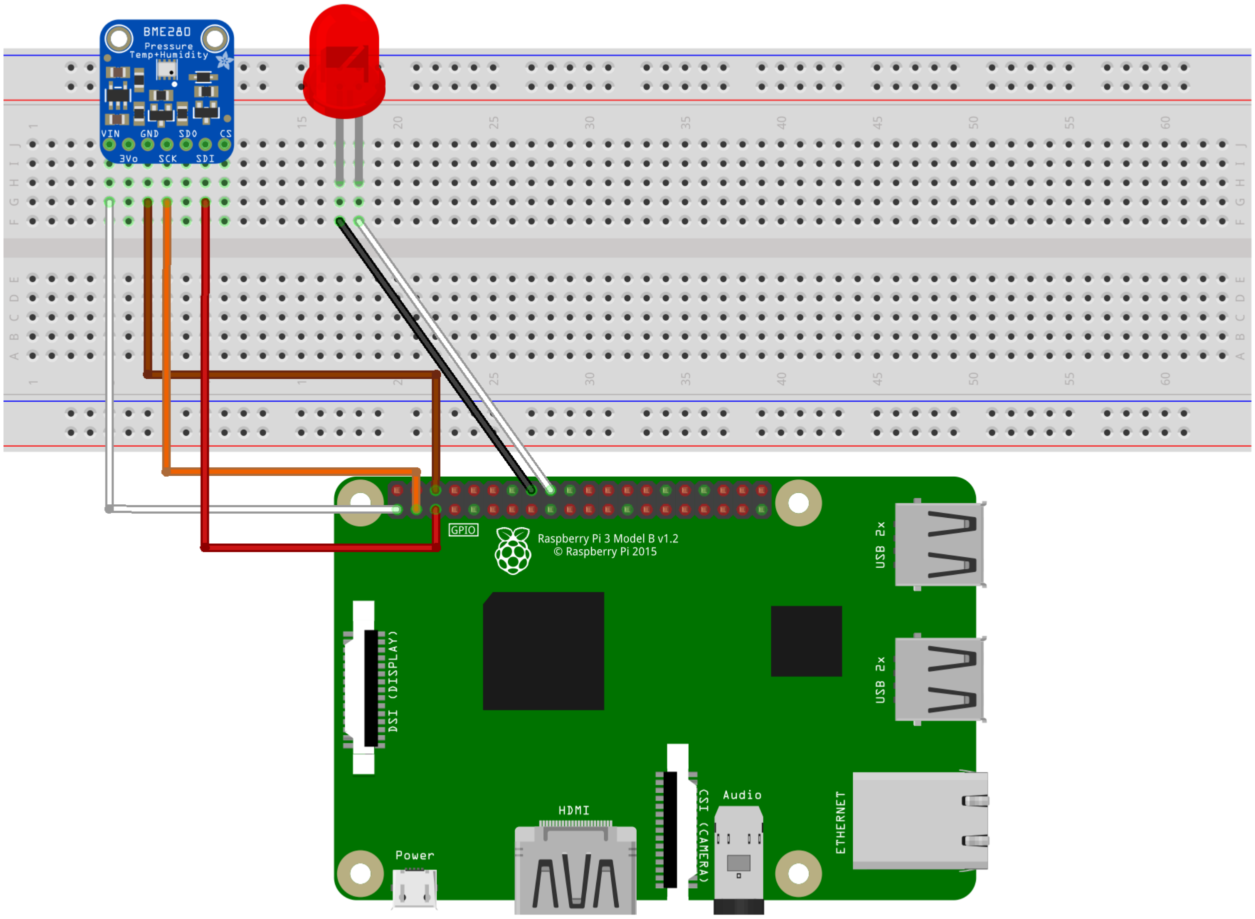
IOT PHASE 3

SMART PARKING

DEVELOPMENT PART 1

Building a camera-based parking detection system with Raspberry Pi and Microsoft Azure involves multiple steps, both in terms of hardware setup and software development. I'll break down the project into a detailed, step-by-step guide:



1. Project Planning and Requirements Gathering:

1. Determine the exact requirements:

  - Number of parking spots to monitor.

- The frequency of parking status updates.

- Desired accuracy of the system.

2. Identify stakeholders and gather feedback:

- Parking lot management.

- Regular users or drivers.

2. Hardware Procurement:

1. Purchase necessary hardware components:

- Raspberry Pi (preferably a Pi 3 or Pi 4).

- Raspberry Pi Camera Module.

- SD card with Raspbian OS installed.

- Power source for Raspberry Pi.

2. Setup of the Raspberry Pi Online Simulator:

- Go to the Raspberry Pi Online Simulator.

- On the right side, you'll see a simulated breadboard and Raspberry Pi, and on the left, you'll have a coding area.

- Since we don't have a real camera, replace the sample code with a simple Python script that simulates the sending of parking spot statuses.

3. Hardware Setup:

1. Connect the Raspberry Pi Camera Module to the Pi.

2. Set up the Raspberry Pi with monitor, keyboard, and mouse.

3. Power on and ensure the camera is correctly recognized.

4. Software Setup on Raspberry Pi:

1. Update the Pi and install necessary libraries:

```bash
sudo apt-get update

sudo apt-get upgrade

sudo apt-get install python3-picamera python3-opencv
```

~~~
```
~~~

2. Install Azure IoT SDK for Python:

```bash
pip install azure-iot-device
```

5. Microsoft Azure Setup:

1. Create an Azure account if you haven't.

2. Set up an IoT Hub:

   - Navigate to the Azure portal.

   - Create a new IoT Hub instance.

3. Device Registration:

   - Register your Raspberry Pi with the IoT Hub.

   - Note the device connection string for later use.

6. Developing the Parking Detection System:

1. Write a Python script on the Raspberry Pi:

   - Capture images using the Pi camera.

   - Process images using OpenCV for parking spot occupancy detection (e.g., through image differencing).

   - Send parking status to Azure IoT Hub.

2. Test the script locally on the Raspberry Pi to ensure correct functionality.

7. Integration with Microsoft Azure:

1. Modify the Python script to send messages to Azure IoT Hub using the device connection string.

2. On Azure, set up Stream Analytics to process incoming data and store it, or trigger any necessary actions based on the parking status.

3. Optionally, integrate with other Azure services:

   - Azure Functions for serverless actions.

  - Azure Logic Apps for workflow automation.

  - Azure Maps for visually representing parking spots on a map.

8. Deployment and Testing:

1. Deploy the camera and Raspberry Pi in the parking area for real-world testing.

2. Analyze results in Azure and verify the system's accuracy.

3. Make necessary adjustments to the image processing algorithm based on real-world results.

9. Feedback Loop and Iterations:

1. Collect feedback from users about the system's accuracy and usability.

2. Implement improvements based on feedback and retest.

10. Scaling and Maintenance:

1. Scale the solution, if needed, by adding more cameras and Raspberry Pis for larger parking areas.

2. Regularly maintain the system:

  - Clean camera lenses.

  - Check for software updates.

  - Ensure Azure costs are within budget.

11. Documentation and Training:

1. Document the system setup, software details, and any troubleshooting steps.

2. Train parking lot staff or management on how to use the system.

Code:

```
import picamera

def capture_image():

    with picamera.PiCamera() as camera:

        camera.resolution = (1280, 720)

        image_path = 'parking_image.jpg'
        camera.capture(image_path)
    return image_path
import cv2
import numpy as np
empty_space_image = cv2.imread('empty_parking.jpg', 0)
def is_parking_occupied(current_image_path):
    current_image = cv2.imread(current_image_path, 0)
    difference = cv2.absdiff(empty_space_image, current_image)
    _, thresholded = cv2.threshold(difference, 25, 255, cv2.THRESH_BINARY)
    occupied_pixels = np.sum(thresholded == 255)
    total_pixels = thresholded.size
    return (occupied_pixels / total_pixels) > 0.05
pip install azure-iot-device
from azure.iot.device import IoTHubDeviceClient, Message
CONNECTION_STRING = "Your_Device_Connection_String"
def send_to_azure(status):
    client = IoTHubDeviceClient.create_from_connection_string(CONNECTION_STRING)
    message = Message(status)
    client.send_message(message)
def main():
    while True:
        image_path = capture_image()
        if is_parking_occupied(image_path):
            send_to_azure("occupied")
            print("Parking spot occupied!")
        else:
            send_to_azure("available")
            print("Parking spot available!")
```