Smart door opening system

➢ Project Requirements

Hardware Components:

- QR Code Scanner: For scanning QR codes.
- Microcontroller: Raspberry Pi.
- Relay Module: Controls the door lock mechanism.
- Power Supply: Ensure it matches the requirements of your microcontroller and the lock.

Software Tools:

- Programming Languages: Python (for Raspberry Pi).
- Libraries:

  RPi.GPIO : For GPIO control.

  time : For delays and timing.

  Pyqrcode : To generate QR codes.

  Pillow(PIL) : To display QR codes.

➢ Installation Steps

1. Install Python and Required Tools

Verify Python is installed (Python 3 is recommended):

```
>>>  python3 --version
```

Install pip (Python package manager):

```
>>> apt install python3-pip
```

2. Install Required Libraries

- RPi.GPIO (GPIO library for hardware control):

  ```
  >>> pip3 install RPi.GPIO
  ```

- time(Default Python library): No installation is required as it is part of the Python standard library.

- pyqrcode* (QR code generation library):

```
>>> pip3 install pyqrcode
```

- Pillow (PIL)* (Image handling library):

```
>>>pip3 install pillow
```

3. Configure GPIO and Enable I2C (if needed)

Open the Raspberry Pi configuration tool:

```
>>>raspi-config
```

2. Enable GPIO and other interfaces (if not already enabled).

4. Test the Installation

Run a simple test to ensure libraries are installed:

```
python3
>>> import RPi.GPIO
>>> import pyqrcode
>>> from PIL import Image
>>> exit()
```

2. If no errors occur, the setup is successful.

➢ Execution Steps

Step 1: Generate and Store the QR Codes

- Use the pyqrcode library to generate QR codes with links or data.
- Save the generated QR codes as PNG files in a designated directory .

Step 2: Program the Microcontroller

- Connect the Raspberry Pi to your development environment (keyboard, mouse, or SSH).
- Copy the Python script to the Raspberry Pi and ensure all libraries are installed
- Test hardware connections:
  - Verify GPIO pin configurations.
  - Ensure the stepper motor, ultrasonic sensor, LEDs, and switches respond correctly.

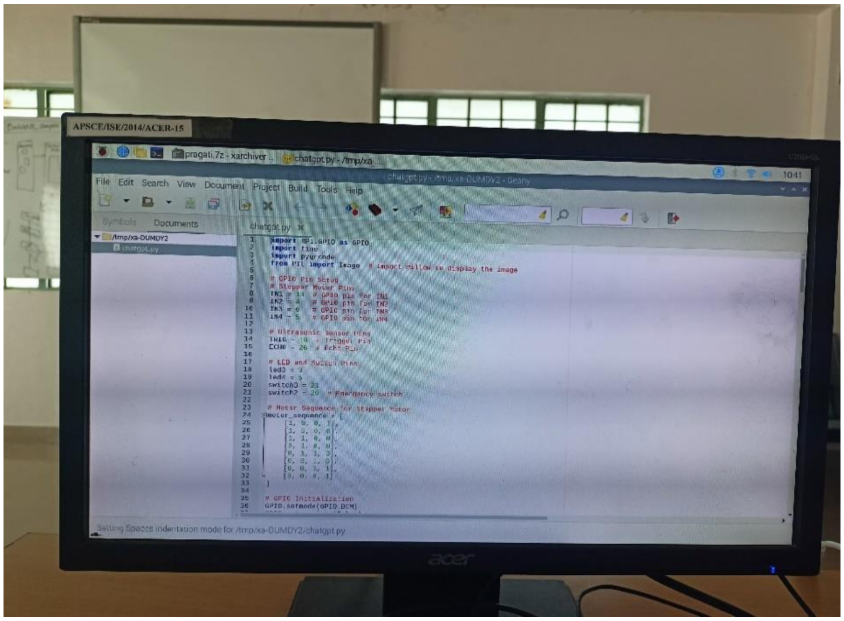
Step 3: Testing and Deployment

- Initial Testing:
  - Run the script and simulate different scenarios (e.g., object detection, emergency button press).
  - Verify QR code generation and form submission workflow.

- Debugging:
  - Address any hardware or software issues during the testing phase.

- Deployment:
  - Install the Raspberry Pi in the intended environment.
  - Securely connect all hardware components.
  - Run the script continuously for real-world operation.

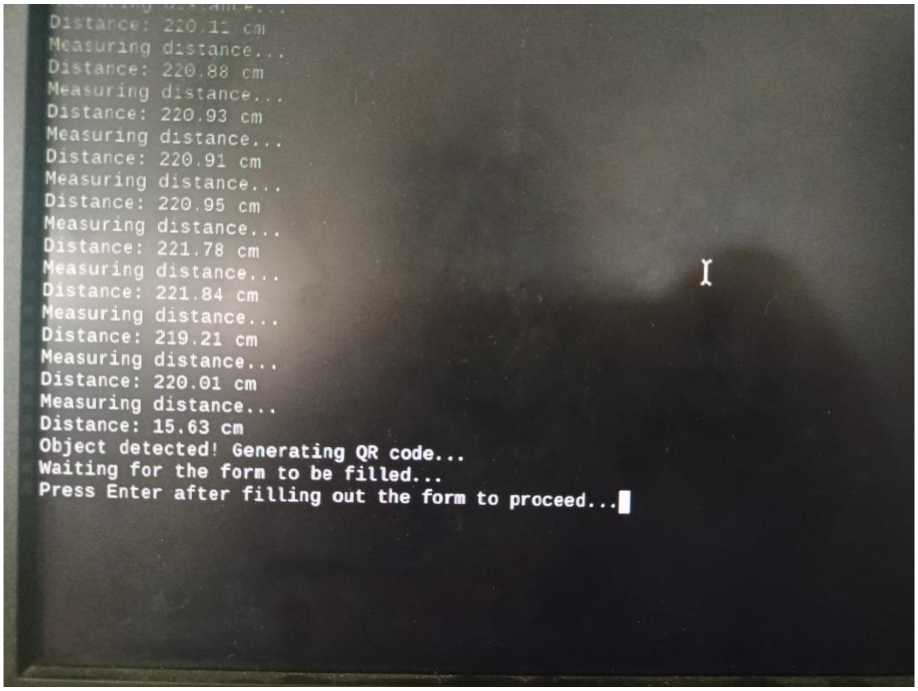
➢ Snapshots


APSCE/ISE/2014/ACER-15
pragati.7z - xarchiver ...
chatgpt.py - /tmp/xa ...
chatgpt.py - /tmp/xa-DUMDY2 - Geany
File Edit Search View Document Project Build Tools Help
Symbols Documents
/tmp/xa-DUMDY2
chatgpt.py
10:41
acer

```
import RPi.GPIO as GPIO
import time
import pyqrcode
from PIL import Image  # import Pillow to display the image

# GPIO Pin Setup
# Stepper Motor Pins
IN1 = 17  # GPIO pin for IN1
IN2 = 4   # GPIO pin for IN2
IN3 = 6   # GPIO pin for IN3
IN4 = 5   # GPIO pin for IN4

# Ultrasonic Sensor Pins
TRIG = 19  # Trigger Pin
ECHO = 26  # Echo Pin

# LCD and Switch Pins
led3 = 9
led4 = 6
switch3 = 21
switch2 = 20  # Emergency switch

# Motor Sequence for Stepper Motor
motor_sequence = [
    [1, 0, 0, 1],
    [1, 0, 0, 0],
    [1, 1, 0, 0],
    [0, 1, 0, 0],
    [0, 1, 1, 0],
    [0, 0, 1, 0],
    [0, 0, 1, 1],
    [0, 0, 0, 1]
]

# GPIO Initialization
GPIO.setmode(GPIO.BCM)
```

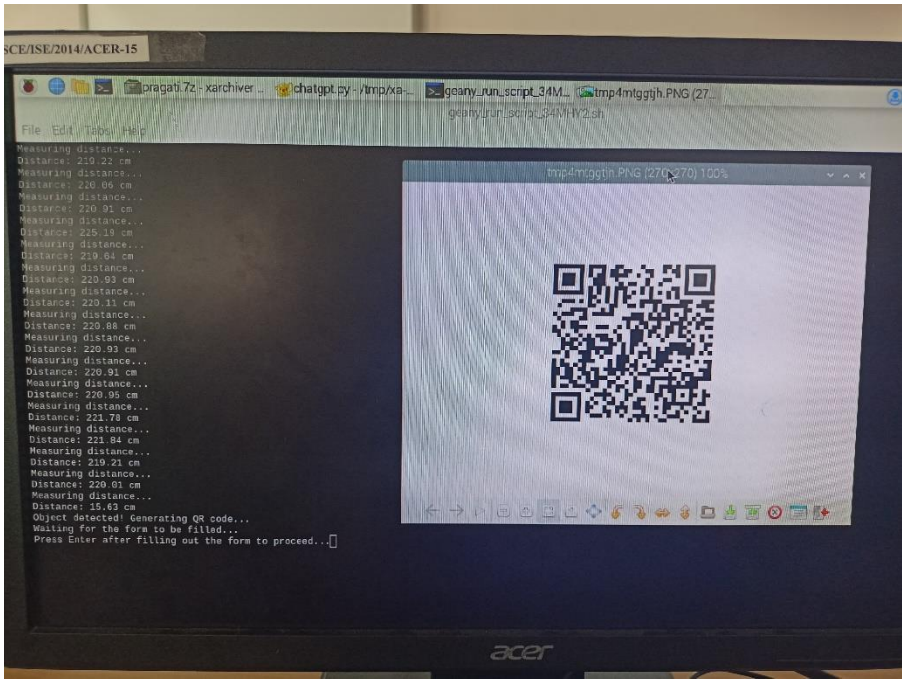
Setting Spaces indentation mode for /tmp/xa-DUMDY2/chatgpt.py

The main code of the project.



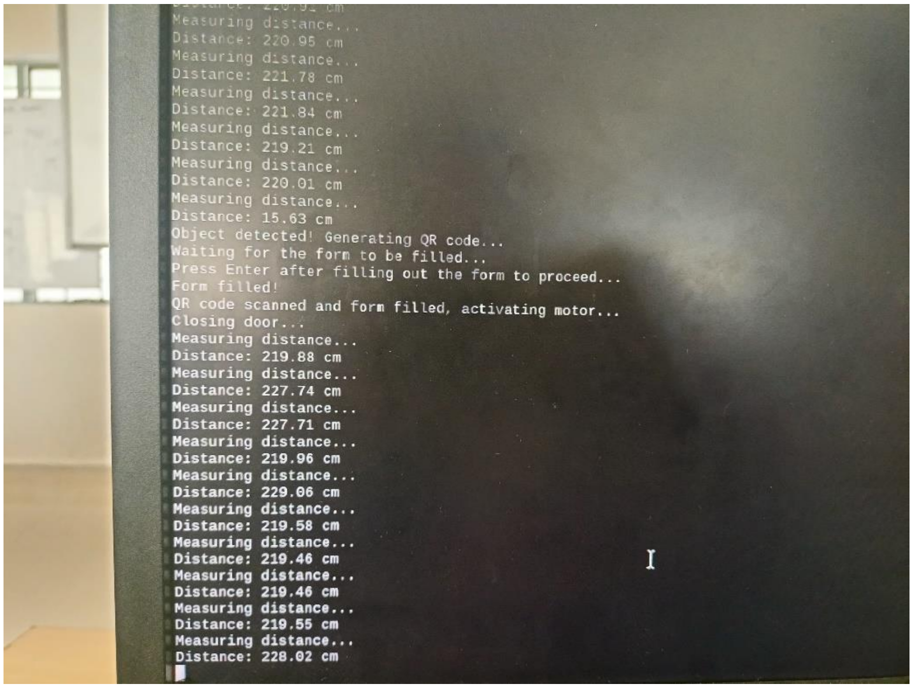
```
Distance: 220.11 cm
Measuring distance...
Distance: 220.88 cm
Measuring distance...
Distance: 220.93 cm
Measuring distance...
Distance: 220.91 cm
Measuring distance...
Distance: 220.95 cm
Measuring distance...
Distance: 221.78 cm
Measuring distance...
Distance: 221.84 cm
Measuring distance...
Distance: 219.21 cm
Measuring distance...
Distance: 220.01 cm
Measuring distance...
Distance: 15.63 cm
Object detected! Generating QR code...
Waiting for the form to be filled...
Press Enter after filling out the form to proceed...
```

The output console before filling the form.



```
Measuring distance...
Distance: 219.22 cm
Measuring distance...
Distance: 220.06 cm
Measuring distance...
Distance: 220.91 cm
Measuring distance...
Distance: 225.19 cm
Measuring distance...
Distance: 219.64 cm
Measuring distance...
Distance: 220.93 cm
Measuring distance...
Distance: 220.11 cm
Measuring distance...
Distance: 220.88 cm
Measuring distance...
Distance: 220.93 cm
Measuring distance...
Distance: 220.91 cm
Measuring distance...
Distance: 220.95 cm
Measuring distance...
Distance: 221.78 cm
Measuring distance...
Distance: 221.84 cm
Measuring distance...
Distance: 219.21 cm
Measuring distance...
Distance: 220.01 cm
Measuring distance...
Distance: 15.63 cm
Object detected! Generating QR code...
Waiting for the form to be filled...
Press Enter after filling out the form to proceed...
```

Generating QR code and waiting for the form to be filled.



```
Distance: 220.91 cm
Measuring distance...
Distance: 220.95 cm
Measuring distance...
Distance: 221.78 cm
Measuring distance...
Distance: 221.84 cm
Measuring distance...
Distance: 219.21 cm
Measuring distance...
Distance: 220.01 cm
Measuring distance...
Distance: 15.63 cm
Object detected! Generating QR code...
Waiting for the form to be filled...
Press Enter after filling out the form to proceed...
Form filled!
QR code scanned and form filled, activating motor...
Closing door...
Measuring distance...
Distance: 219.88 cm
Measuring distance...
Distance: 227.74 cm
Measuring distance...
Distance: 227.71 cm
Measuring distance...
Distance: 219.96 cm
Measuring distance...
Distance: 229.06 cm
Measuring distance...
Distance: 219.58 cm
Measuring distance...
Distance: 219.46 cm
Measuring distance...
Distance: 219.46 cm
Measuring distance...
Distance: 219.55 cm
Measuring distance...
Distance: 228.02 cm
```

After scanning QR code and form filled, activating motor. And the cycle repeats.