Smart restroom on automatic room light sensor

Creating a platform for water overflow control in a tank using MIT App Inventor involves several steps. MIT App Inventor is a user-friendly platform for creating Android apps, and it can be used to build a mobile app that monitors and controls water levels in a tank. Here's a basic outline of the steps involved.

Design app interference

• Create two labels (on off light and status).

• Add a button . Block-Based Coding:

• Use MIT App Inventor's block-based coding to program the app:

• When Screen1.Initialize:

• Initialize any necessary variables .

• When Button.Click:

• Toggle the state of the room free or valve (Open/Closed) .

• Update the status label accordingly .

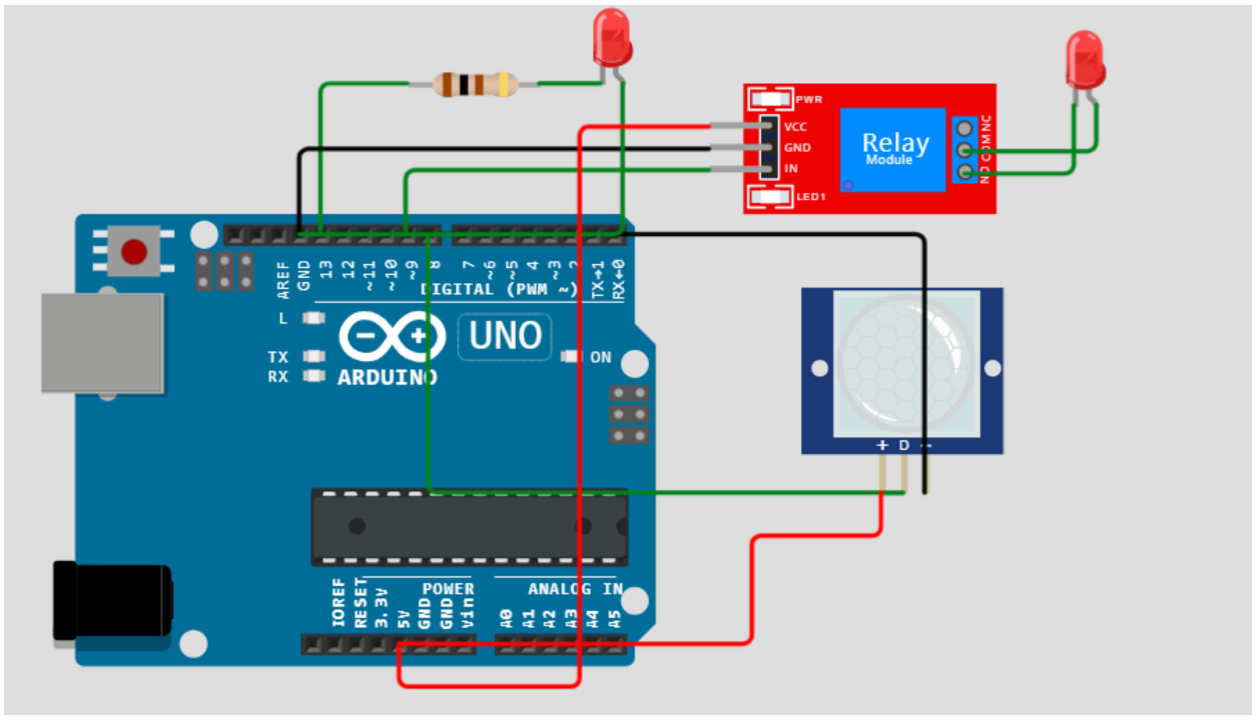
• Use a Timer component to periodically check the vacency(simulated for this example).

• Inside the Timer event handler, you can generate vacancy values as follows:

• Set a variable to a random number within a range representing the restroom.

• Display the vacency on the appropriate label

Circuit design & program



Input program:

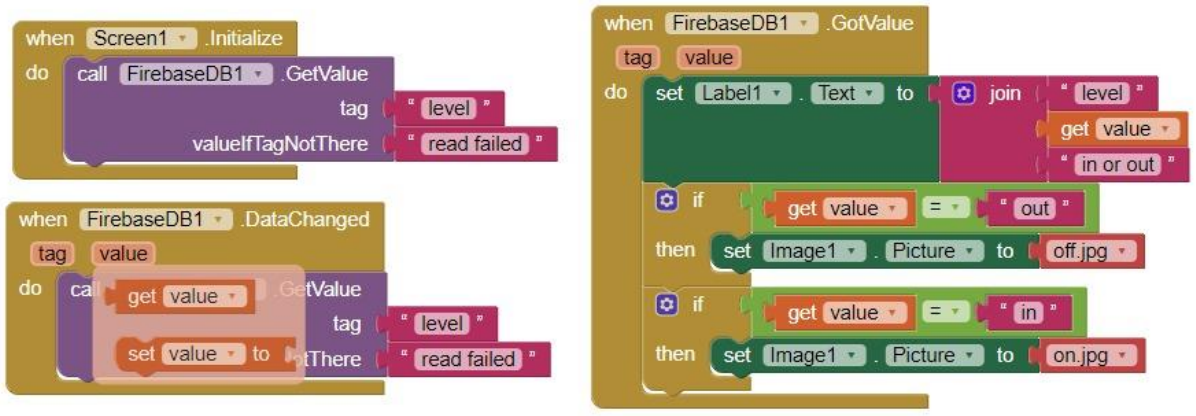
Python

```
import time

import RPi.GPIO as GPIO



in1 = 9

sensor = 8

led = 13
```

```
t = 0

GPIO.setmode(GPIO.BCM)
GPIO.setup(in1, GPIO.OUT)
GPIO.setup(sensor, GPIO.IN)
GPIO.setup(led, GPIO.OUT)

GPIO.output(in1, GPIO.HIGH)
GPIO.output(led, GPIO.LOW)

while time.time() < 13:
    GPIO.output(led, GPIO.HIGH)
    time.sleep(0.05)
    GPIO.output(led, GPIO.LOW)
    time.sleep(0.05)
GPIO.output(led, GPIO.LOW)

while True:
    GPIO.output(in1, GPIO.HIGH)
    GPIO.output(led, GPIO.LOW)

    if GPIO.input(sensor) == GPIO.HIGH:
        t = int(time.time() * 1000)
```

```
while time.time() * 1000 < (t + 5000):
    GPIO.output(in1, GPIO.LOW)
    GPIO.output(led, GPIO.HIGH)

    if time.time() * 1000 > (t + 2300) and GPIO.input(sensor) == GPIO.HIGH:
        t = int(time.time() * 1000)
```

Input program:



```
when Screen1 .Initialize
do  call FirebaseDB1 .GetValue
                         tag  "level"
          valueIfTagNotThere  "read failed"

when FirebaseDB1 .DataChanged
  tag  value
do  call  get value   .GetValue
                         tag  "level"
    set value to       tThere  "read failed"

when FirebaseDB1 .GotValue
  tag  value
do  set Label1 . Text to  join  "level"
                                get value
                                "in or out"
    if    get value = "out"
    then  set Image1 . Picture to  off.jpg
    if    get value = "in"
    then  set Image1 . Picture to  on.jpg
```

Creating a platform for vaceny control in a restroom using MIT App Inventor involves several steps. MIT App Inventor is a user-friendly platform for creating Android apps, and it can be used to build a mobile app that monitors and controls vacancy in a restroom. Here's a basic outline of the steps involved:

1. Define the Requirements:

Clearly define the requirements and functionalities of your vacancy control system. You'll need to specify what features you want in your app, such as monitoring vacancy , sending alerts, and controlling restroom or valves.

2. Design the User Interface:

Use MIT App Inventor's drag-and-drop interface to design the user interface for your app. Create screens for monitoring vacancy , sending notification , and controlling the system.

3. Connect to Hardware:

- If you're connecting the app to hardware like sensors or actuators, you'll need to choose appropriate components and interface with them using Bluetooth, Wi-Fi, or other communication protocols

4. Coding in Blocks:

- Use the MIT App Inventor's visual programming language, which is block-based, to write the code for your app. This includes defining how the app interacts with the hardware, processes sensor data, and controls devices.

5. Set Up Alerts:

- Implement a notification system that sends alerts (push notifications, SMS, or email) to the user's device if the water level exceeds a predefined limit.

6. Testing and Debugging:

- Test the app thoroughly to ensure that it functions as expected. Debug any issues that may arise during testing

7. User Authentication (Optional):

- If necessary, implement user authentication to restrict access to the app and its control features

8. Documentation and Help:

- Create user documentation or help sections within the app to guide users on how to use it effectively

9. App Deployment:

- Once the app is ready, you can export it as an Android application and distribute it to your intended users. You can publish it on the Google Play Store or share the APK file directly.

10. Maintenance and Updates:

- Continue to maintain and update the app as needed, addressing any user feedback or bug reports. Remember that building an app to control physical systems requires careful consideration of safety and reliability.

It's essential to have a good understanding of the hardware you're connecting to and to thoroughly test the system before deploying it in a real-world environment. Additionally, consider the power source and connectivity options for your monitoring and control components