SMART WATER MANAGEMENT

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 (for displaying the water level and status).
- Add a button (for simulating a pump or valve control).

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 pump or valve (Open/Closed)

.

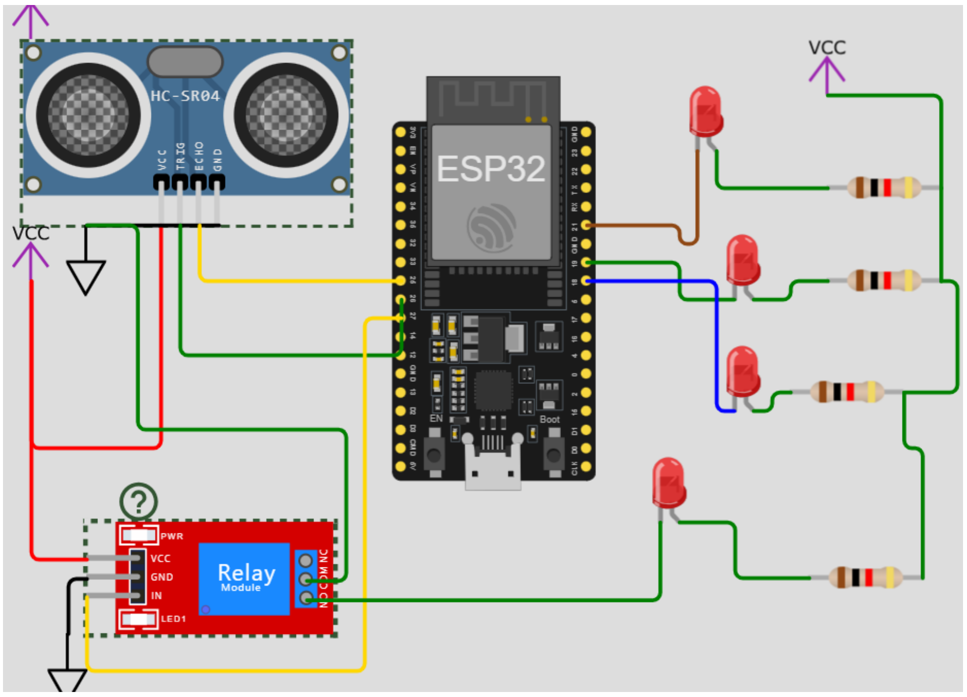
- Update the status label accordingly

.

- Use a Timer component to periodically check the water level (simulated for this example).
- Inside the Timer event handler, you can generate random water level values as follows:
- Set a variable to a random number within a range representing the water level.
- Display the water level on the appropriate label.

- Add logic to check if the water level exceeds a predefined threshold.
- If the water level is above the threshold, change the status label to "Overflow Alert!" and send a notification to the user.
- To simulate the notification, you can use the Notifier component to show an alert message when the threshold is crossed.

Circuit design & program



Input program:

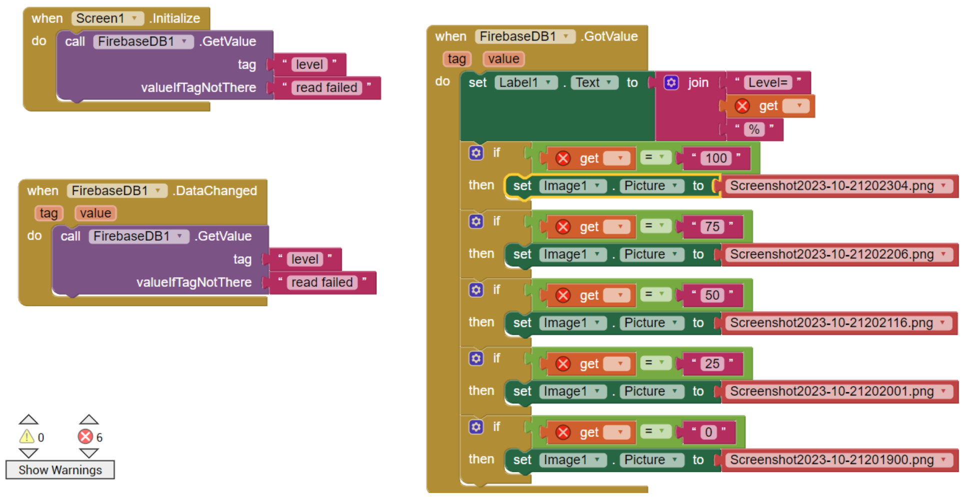
```
#define PIN_TRIG 26

#define PIN_ECHO 25

#define LOWLED   18

#define LOWLED   19

#define LOWLED   21
```

```
#define MOTOR    27

#define FIREBASE_HOST "AIzaSyCt6rGHNysvVaRxodCW2qfWZrQaBdZXFxo"

#define         FIREBASE_AUTH         "https://water-level-control-84a1c-default-
rtdb.firebaseio.com"

using int level=0;



void setup() {



  pinMode(LOWLED,OUTPUT);

  pinMode(MIDLED,OUTPUT);

  pinMode(HIGHLED,OUTPUT);

  pinMode(MOTOR,OUTPUT);

  digitalWrite(LOWLED,HIGH);

  digitalWrite(MIDLED,HIGH);

  digitalWrite(HIGHLED,HIGH);

  digitalWrite(MOTOR,LOW);



  Serial.begin(115200);

  pinMode(PIN_TRIG, OUTPUT);

  pinMode(PIN_ECHO, INPUT);

}



void loop() {

  // Start a new measurement:

  digitalWrite(PIN_TRIG, HIGH);

  delayMicroseconds(10);

  digitalWrite(PIN_TRIG, LOW);
```

```
// Read the result:
int duration = pulseIn(PIN_ECHO, HIGH);
Serial.print("Distance in CM: ");
Serial.println(duration / 58);
Serial.print("Distance in inches: ");
Serial.println(duration / 148);


level = duration / 58;


if(level < 100)
{


     digitalWrite(LOWLED,HIGH);
     digitalWrite(MOTOR,LOW);
     digitalWrite(HIGHLED,HIGH);
     digitalWrite(MIDLED, HIGH);


}
 else if ((level > 200) && (level < 400))
{
     digitalWrite(LOWLED,HIGH);
     digitalWrite(HIGHLED,HIGH);
     digitalWrite(MIDLED,LOW);
```

```
  }
   else if (level >= 400)
  {
       digitalWrite(HIGHLED,LOW);
       digitalWrite(MIDLED,HIGH);
       digitalWrite(LOWLED, HIGH);
       digitalWrite(MOTOR,LOW);
  }


  delay(1000);
}
```

Block design for app



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

when FirebaseDB1 .DataChanged
 tag   value
do  call FirebaseDB1 .GetValue
                     tag  " level "
      valueIfTagNotThere  " read failed "

when FirebaseDB1 .GotValue
 tag   value
do  set Label1 . Text to  join  " Level= "
                                get
                                " % "
    if   get  =  " 100 "
    then set Image1 . Picture to  Screenshot2023-10-21202304.png
    if   get  =  " 75 "
    then set Image1 . Picture to  Screenshot2023-10-21202206.png
    if   get  =  " 50 "
    then set Image1 . Picture to  Screenshot2023-10-21202116.png
    if   get  =  " 25 "
    then set Image1 . Picture to  Screenshot2023-10-21202001.png
    if   get  =  " 0 "
    then set Image1 . Picture to  Screenshot2023-10-21201900.png
```

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:

1. **Define the Requirements:**

- Clearly define the requirements and functionalities of your water overflow control system. You'll need to specify what features you want in your app, such as monitoring water levels, sending alerts, and controlling pumps 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 water levels, setting alarms, 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. **Implement Water Level Monitoring:**

- Use sensors (if applicable) to monitor the water level in the tank. The sensor data may be displayed on the app interface, and you can set up the app to trigger alerts if the water level reaches a certain threshold.

6. **Implement Control Logic:**

- If you want the app to control pumps or valves to prevent overflow, write the logic for this in your app. Make sure to include safety mechanisms to avoid overflows.

7. **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.

8. **Testing and Debugging:**

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

9. **User Authentication (Optional):**

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

10. **Documentation and Help:**

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

11. **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.

12. **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.