Smart Gardening

Luca Avitabile, Johannes Hammerer

Overview

Framework to setup IoT sensors / actuators

Logic defined by user configuration

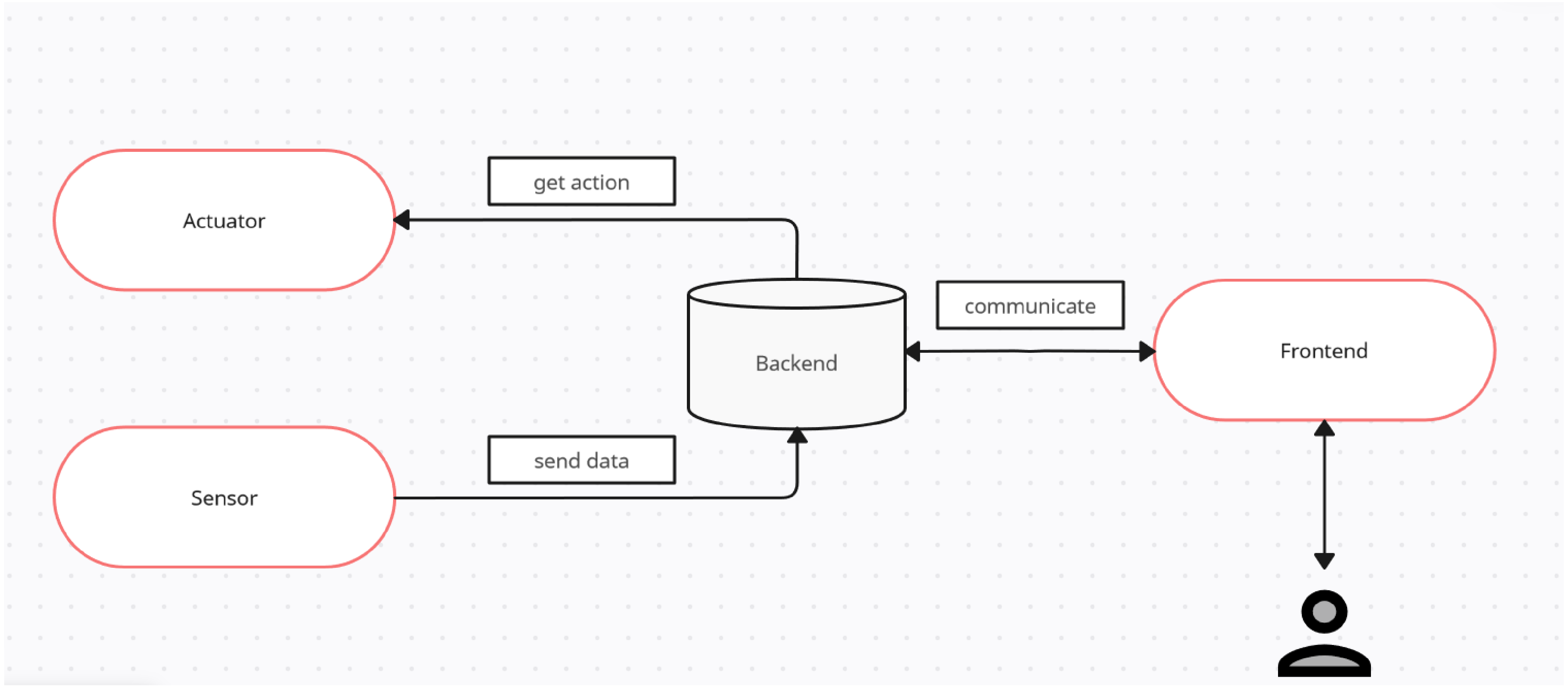
--> Enables multiple sensors and actuators to be connected for different use cases

Concrete Example: Smart Gardening

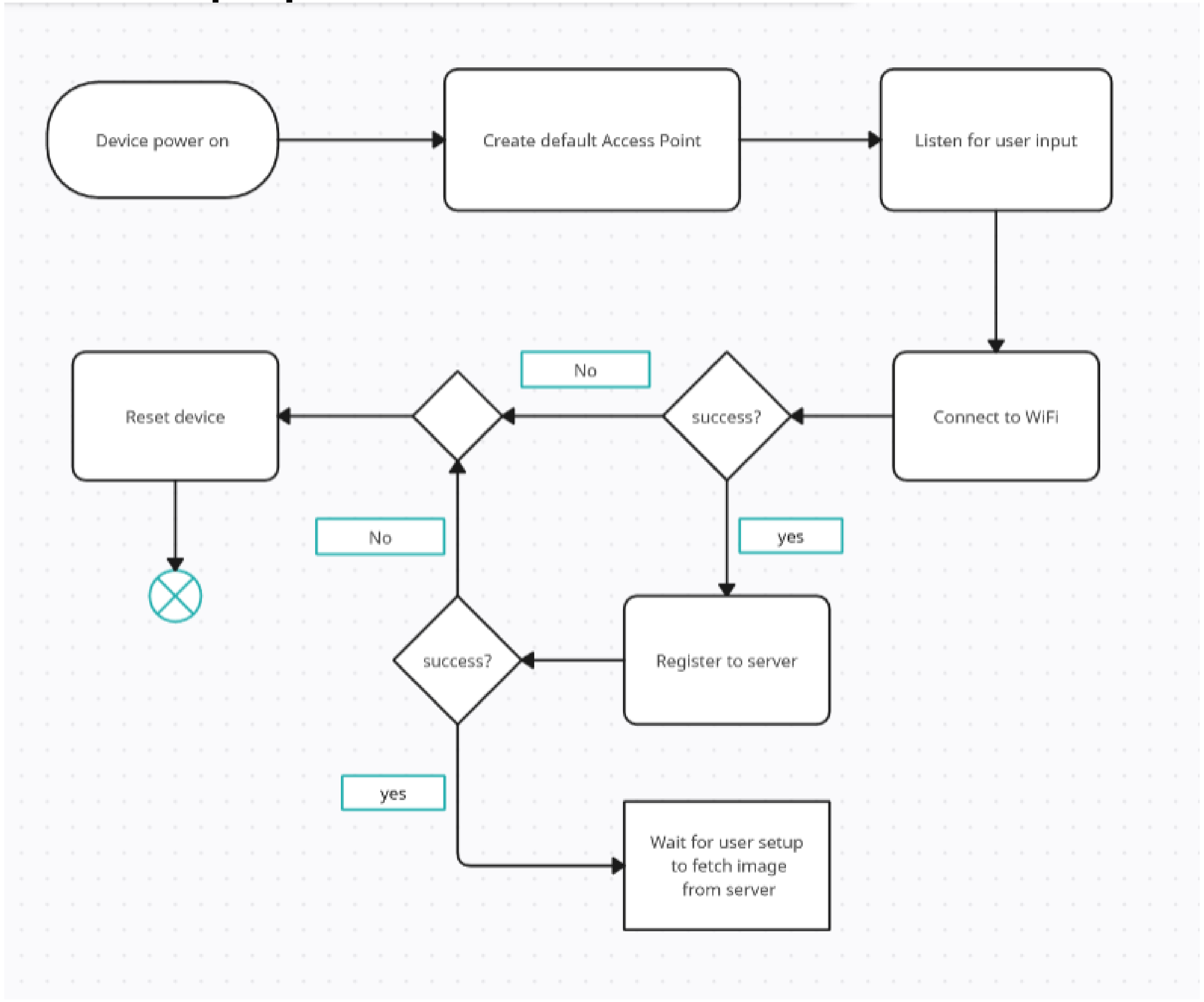
Actuator: Pump

Sensor: Moisture

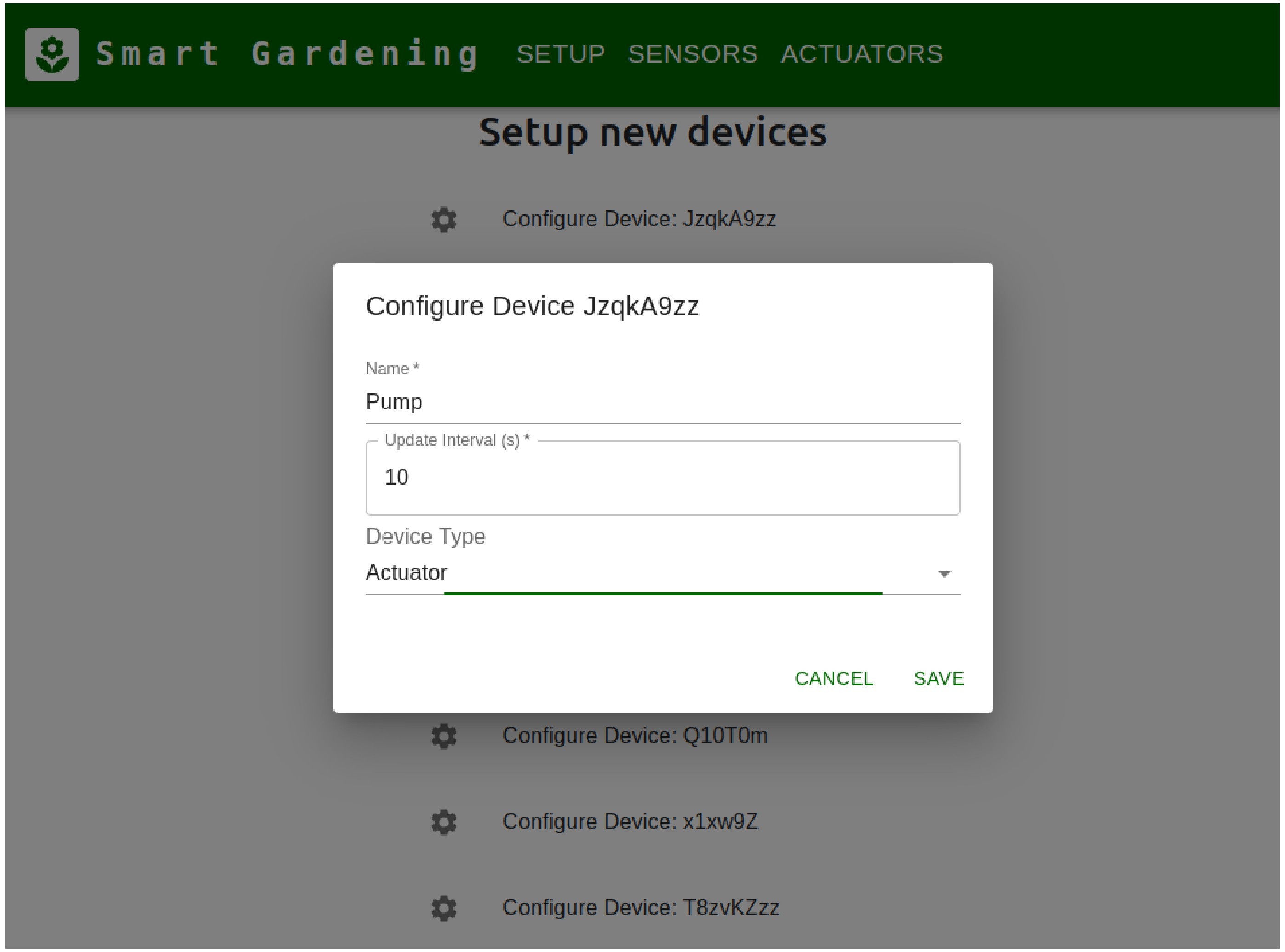
Overview


Actuator
get action
Backend
communicate
Frontend
send data
Sensor

Setup process



Setup process



Setup new devices

Configure Device: JzqkA9zz

Configure Device JzqkA9zz

Name *
Pump

Update Interval (s) *
10

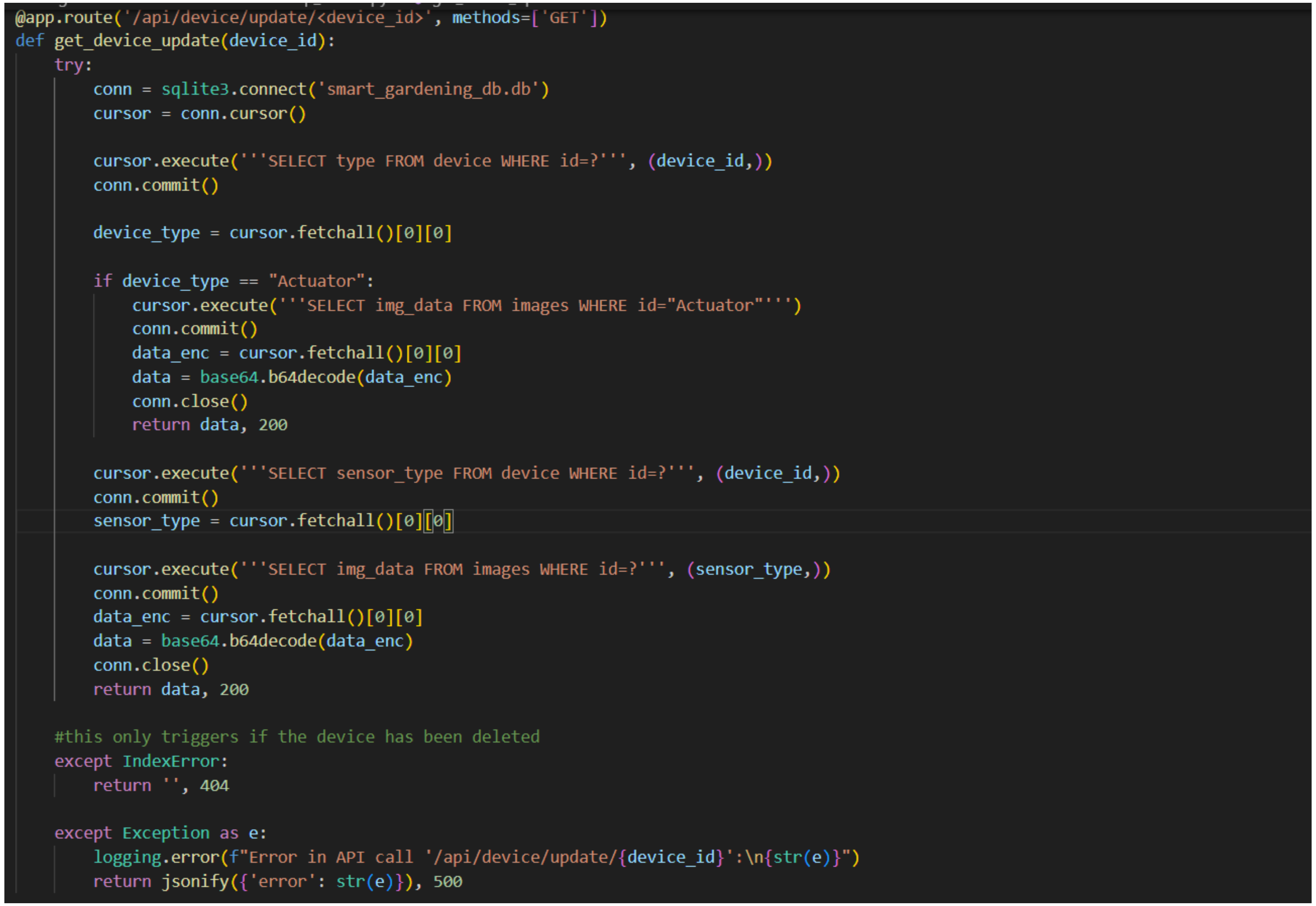
Device Type
Actuator

Configure Device: Q10T0m

Configure Device: x1xw9Z

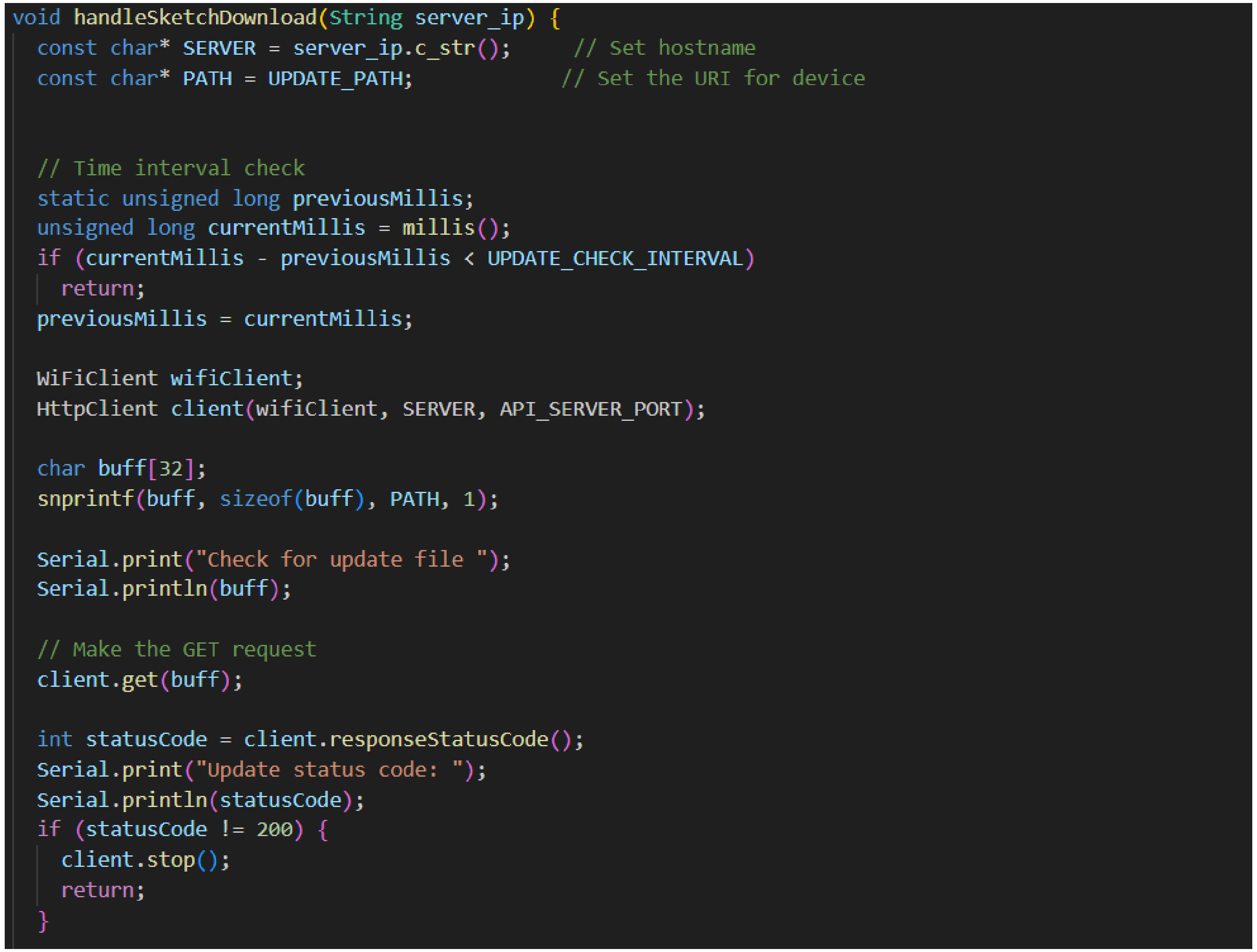
Configure Device: T8zvKZzz

Setup process - Server side



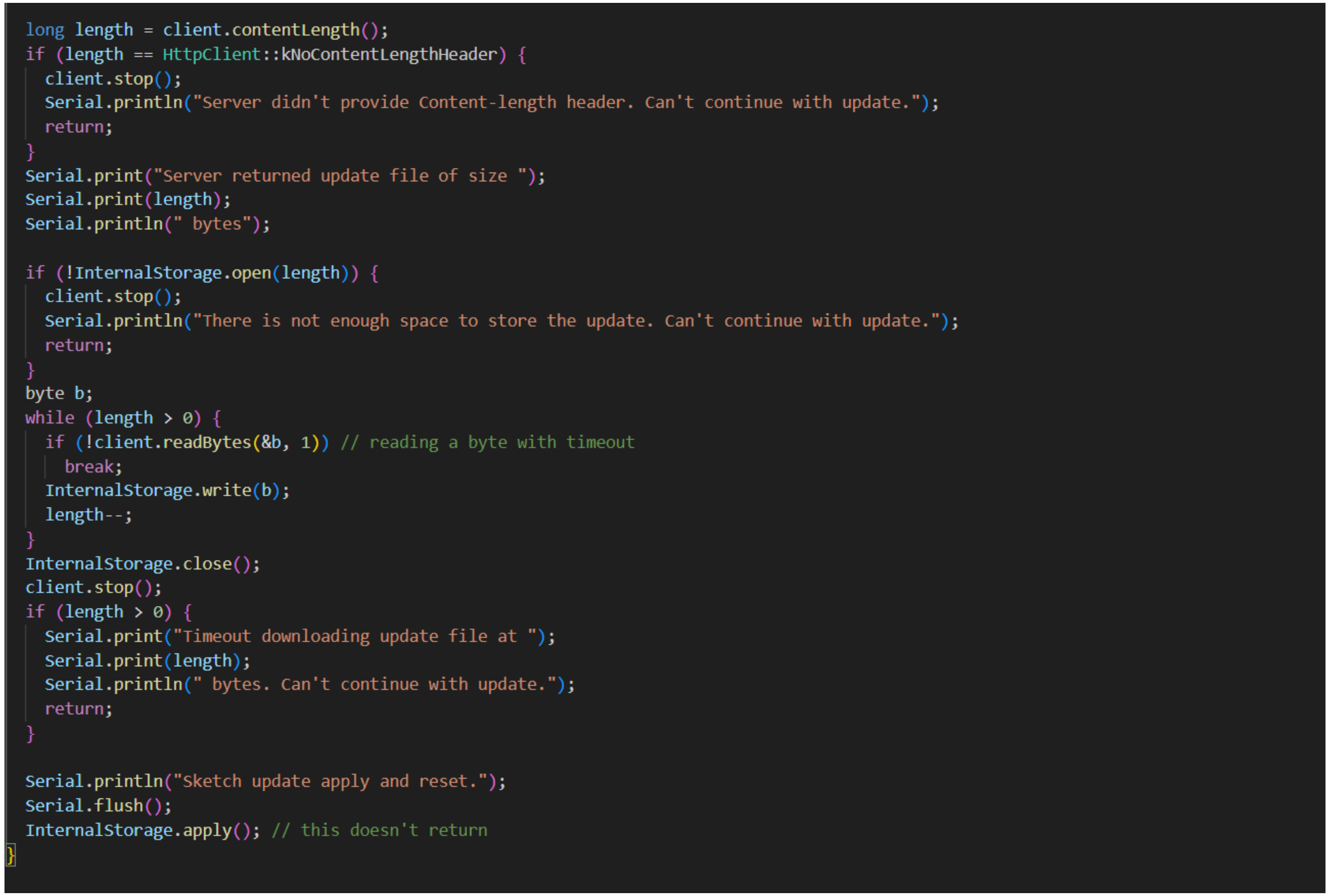
```
@app.route('/api/device/update/<device_id>', methods=['GET'])
def get_device_update(device_id):
    try:
        conn = sqlite3.connect('smart_gardening_db.db')
        cursor = conn.cursor()

        cursor.execute('''SELECT type FROM device WHERE id=?''', (device_id,))
        conn.commit()

        device_type = cursor.fetchall()[0][0]

        if device_type == "Actuator":
            cursor.execute('''SELECT img_data FROM images WHERE id="Actuator"''')
            conn.commit()
            data_enc = cursor.fetchall()[0][0]
            data = base64.b64decode(data_enc)
            conn.close()
            return data, 200

        cursor.execute('''SELECT sensor_type FROM device WHERE id=?''', (device_id,))
        conn.commit()
        sensor_type = cursor.fetchall()[0][0]

        cursor.execute('''SELECT img_data FROM images WHERE id=?''', (sensor_type,))
        conn.commit()
        data_enc = cursor.fetchall()[0][0]
        data = base64.b64decode(data_enc)
        conn.close()
        return data, 200

    #this only triggers if the device has been deleted
    except IndexError:
        return '', 404

    except Exception as e:
        logging.error(f"Error in API call '/api/device/update/{device_id}':\n{str(e)}")
        return jsonify({'error': str(e)}), 500
```

Setup process - Client side



```
void handleSketchDownload(String server_ip) {
  const char* SERVER = server_ip.c_str();     // Set hostname
  const char* PATH = UPDATE_PATH;            // Set the URI for device


  // Time interval check
  static unsigned long previousMillis;
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis < UPDATE_CHECK_INTERVAL)
    return;
  previousMillis = currentMillis;

  WiFiClient wifiClient;
  HttpClient client(wifiClient, SERVER, API_SERVER_PORT);

  char buff[32];
  snprintf(buff, sizeof(buff), PATH, 1);

  Serial.print("Check for update file ");
  Serial.println(buff);

  // Make the GET request
  client.get(buff);

  int statusCode = client.responseStatusCode();
  Serial.print("Update status code: ");
  Serial.println(statusCode);
  if (statusCode != 200) {
    client.stop();
    return;
  }
```

Setup process - Client side

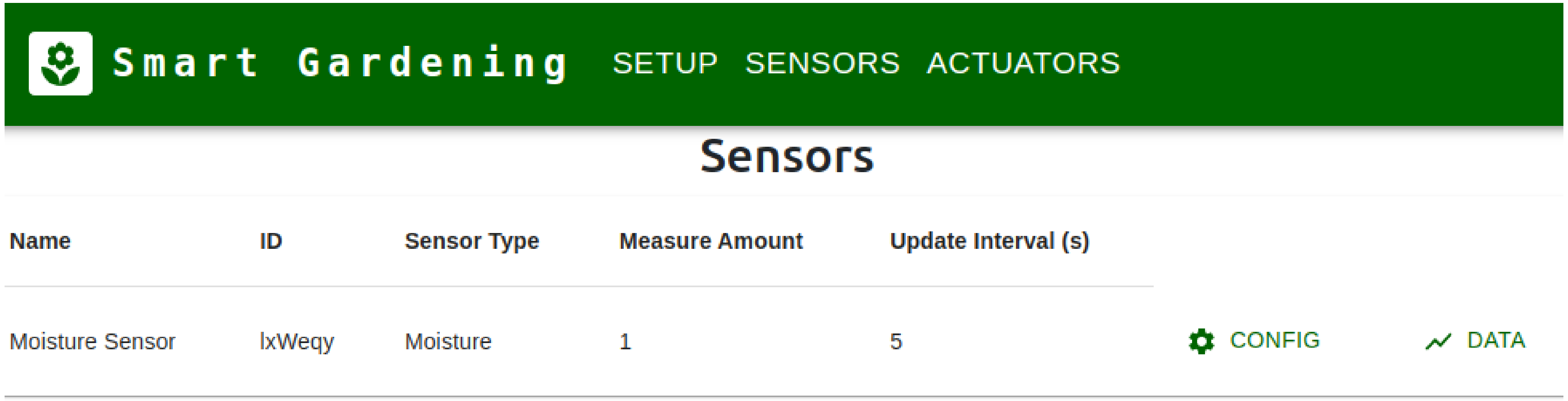


```
long length = client.contentLength();
if (length == HttpClient::kNoContentLengthHeader) {
  client.stop();
  Serial.println("Server didn't provide Content-length header. Can't continue with update.");
  return;
}
Serial.print("Server returned update file of size ");
Serial.print(length);
Serial.println(" bytes");

if (!InternalStorage.open(length)) {
  client.stop();
  Serial.println("There is not enough space to store the update. Can't continue with update.");
  return;
}
byte b;
while (length > 0) {
  if (!client.readBytes(&b, 1)) // reading a byte with timeout
    break;
  InternalStorage.write(b);
  length--;
}
InternalStorage.close();
client.stop();
if (length > 0) {
  Serial.print("Timeout downloading update file at ");
  Serial.print(length);
  Serial.println(" bytes. Can't continue with update.");
  return;
}

Serial.println("Sketch update apply and reset.");
Serial.flush();
InternalStorage.apply(); // this doesn't return
}
```

Sensors

- Send data to backend at defined interval
- Based on data --> activate actuator
- Configurable
- Data for each sensor visible

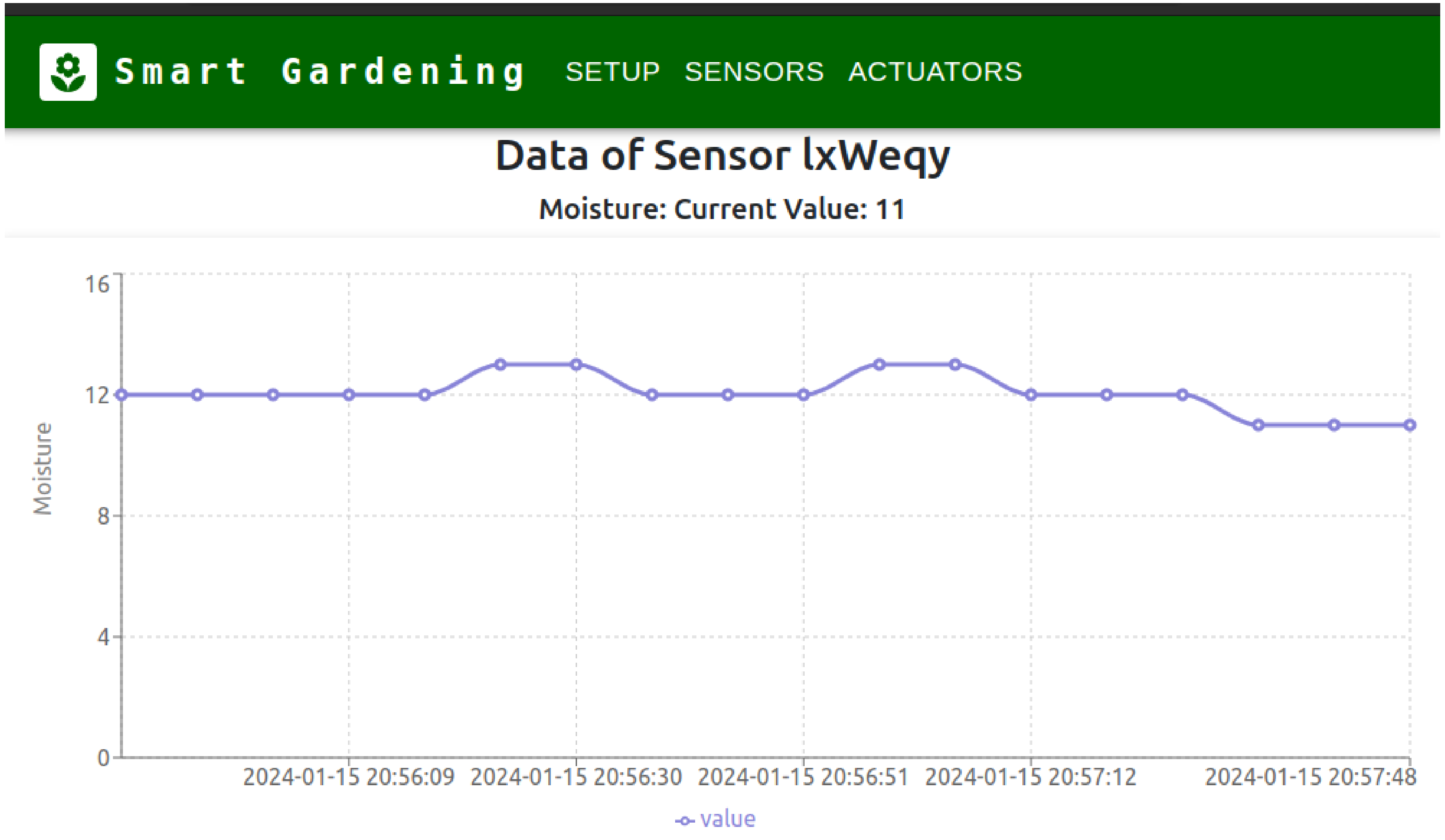
Sensors



Sensors

Name	ID	Sensor Type	Measure Amount	Update Interval (s)		
Moisture Sensor	lxWeqy	Moisture	1	5	CONFIG	DATA

Sensors



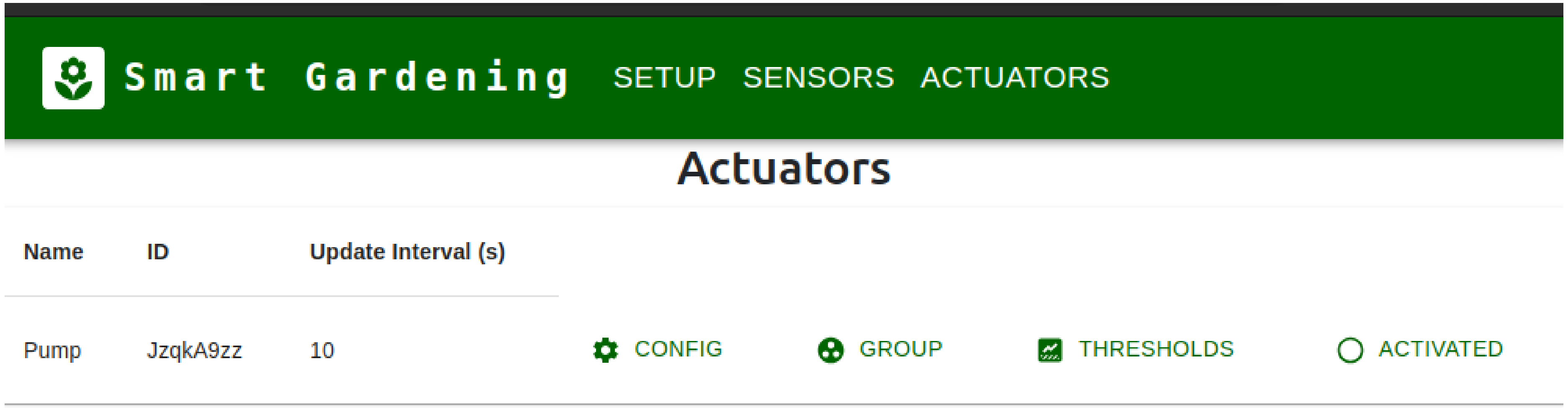
Data of Sensor lxWeqy

Moisture: Current Value: 11

Actuators

- Asks periodically backend if action must be executed
- Configurable
- Manual trigger
  - Based on sensor data that is attached to actuator
- Automatic trigger via sensor data
  - Attach sensors to actuator

Actuators

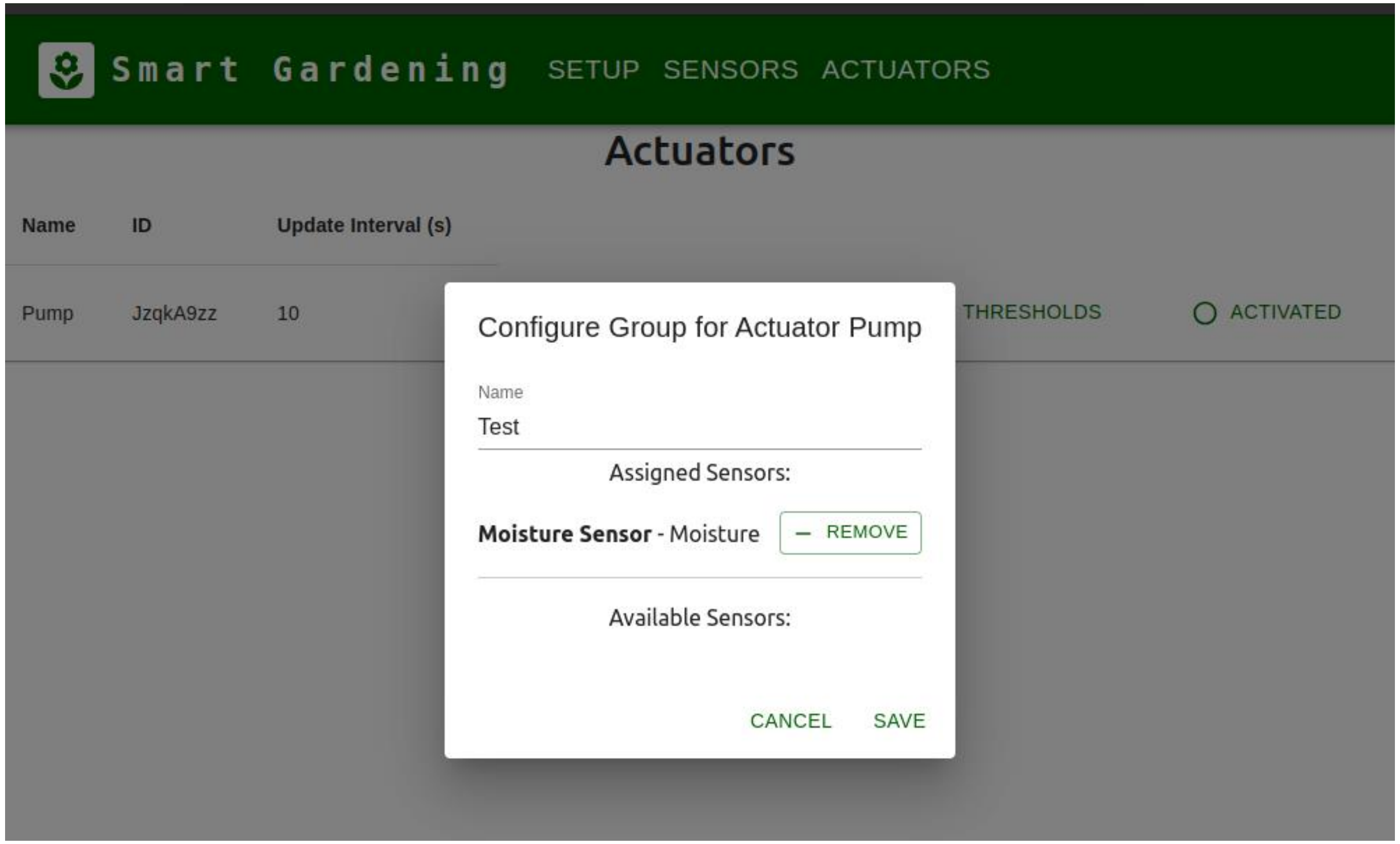


Smart Gardening SETUP SENSORS ACTUATORS

Actuators

Name	ID	Update Interval (s)				
Pump	JzqkA9zz	10	CONFIG	GROUP	THRESHOLDS	ACTIVATED

Actuators



Smart Gardening SETUP SENSORS ACTUATORS

Actuators

Name	ID	Update Interval (s)
Pump	JzqkA9zz	10

THRESHOLDS ACTIVATED

Configure Group for Actuator Pump

Name: Test

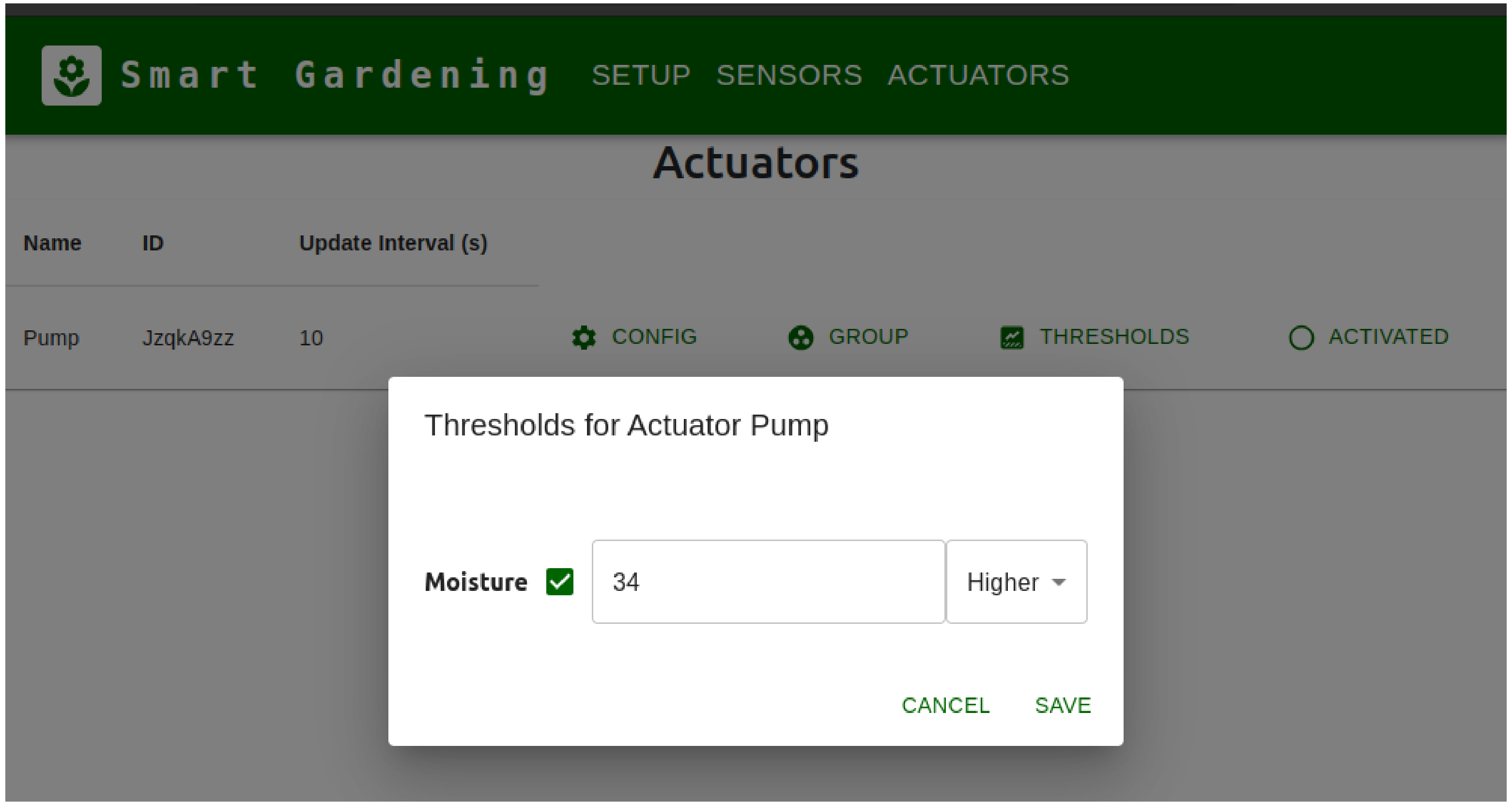
Assigned Sensors:

Moisture Sensor - Moisture — REMOVE

Available Sensors:

CANCEL SAVE

Actuators



Actuators

Name	ID	Update Interval (s)
Pump	JzqkA9zz	10

Thresholds for Actuator Pump

Moisture ✓ 34 Higher

Benefits of this project

- One local centralized hub (backend)
- Images for sensors easy adaptable
- Images for actuators easy adaptable
- --> User has a lot of freedom to implement his own system for specific use-case