Development and Implementation of an IoT System using MQTT and Node-RED

Solar Project

Faculty of Engineering, University of Ruhuna

Table of Contents

[1. IoT System Design and Architecture 4](#_Toc172053735)

[1.1 System Overview 4](#_Toc172053736)

[1.2 Components of the System 4](#_Toc172053737)

[1.2.1 MQTT Protocol 4](#_Toc172053738)

[1.2.2 Node-RED 4](#_Toc172053739)

[1.2.3 Node-RED Dashboards 4](#_Toc172053740)

[1.3 System Architecture Diagram 5](#_Toc172053741)

[1.4 Data Flow and Communication 5](#_Toc172053742)

[2. Implementation 6](#_Toc172053743)

[2.1 Hardware Setup 6](#_Toc172053744)

[2.1.1 Selecting Temperature Sensors 6](#_Toc172053745)

[2.1.2 Sensor Calibration 6](#_Toc172053746)

[2.1.3 Connectivity Setup 6](#_Toc172053747)

[2.2 Software Development 7](#_Toc172053748)

[2.2.1 MQTT Broker Configuration 7](#_Toc172053749)

[2.2.2 Node-RED Flow Design 7](#_Toc172053750)

[2.2.3 Dashboard Creation 7](#_Toc172053751)

[2.3 Integrating Hardware and Software 8](#_Toc172053752)

[3. Budget 9](#_Toc172053753)

[Figure 1.1 IoT system architecture 5](file:///D:\X\project\Other\FYP\IoT%20System.docx#_Toc172054306)

[Figure 2.1 Dashboard layout 8](file:///D:\X\project\Other\FYP\IoT%20System.docx#_Toc172054307)

# 1. IoT System Design and Architecture

## 1.1 System Overview

The IoT monitoring system is designed to collect system data from various sensors, transmit the data to a central server using the MQTT protocol, and visualize the data using Node-RED and Node-RED dashboards. The system aims to provide real-time monitoring and analysis of system data, enabling proactive responses to temperature changes.

## 1.2 Components of the System

### 1.2.1 MQTT Protocol

* **Description**: MQTT (Message Queuing Telemetry Transport) is a lightweight messaging protocol designed for constrained devices and low-bandwidth, high-latency networks. It follows a publish/subscribe model.
* **Features**:
  + Lightweight and efficient
  + Supports Quality of Service (QoS) levels
  + Provides reliable communication
* **MQTT Broker**: The central component that handles message routing between publishers (sensors) and subscribers (Node-RED).

### 1.2.2 Node-RED

* **Description**: Node-RED is a flow-based development tool for visual programming, designed for wiring together hardware devices, APIs, and online services.
* **Capabilities**:
  + Real-time data processing
  + Integration with various protocols and services
  + Easy-to-use graphical interface
* **Usage**: Node-RED is used to design the data flow, process the incoming temperature data, and route it to the dashboard for visualization.

### 1.2.3 Node-RED Dashboards

* **Description**: Node-RED Dashboards are used to create web-based user interfaces to visualize the data processed by Node-RED.
* **Components**:
  + Graphs and charts to display temperature trends
  + Gauges to show real-time temperature values
  + Notifications for alerting abnormal temperature readings
* **Customization**: Dashboards can be customized to meet specific user requirements and provide intuitive data visualization.

## 1.3 System Architecture Diagram

The system architecture diagram illustrates the overall structure and data flow within the IoT temperature monitoring system. It includes the following components:

* Sensors: Collect data
* Microcontroller/Gateway: Interface between sensors and MQTT broker
* MQTT Broker: Manages message routing
* Node-RED: Processes data and manages dashboard
* Node-RED Dashboard: Visualizes data for end-users
* InfuxDB Database: Store data for later analyze

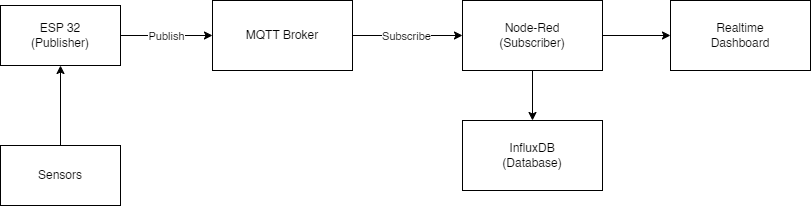


Figure 1. IoT system architecture

## 1.4 Data Flow and Communication

* **Data Collection**: Temperature sensors collect temperature data at regular intervals and send it to the microcontroller.
* **Data Transmission**: The microcontroller publishes the temperature data to the MQTT broker.
* **Data Routing**: The MQTT broker receives the data and routes it to the Node-RED server.
* **Data Processing**: Node-RED processes the incoming data, checks for anomalies, and prepares it for visualization.
* **Data Visualization**: The processed data is sent to the Node-RED dashboard, where it is visualized in real-time through graphs, gauges, and notifications.
* **Alerts and Notifications**: If the temperature readings exceed predefined thresholds, Node-RED triggers alerts and notifications on the dashboard.

# 2. Implementation

## 2.1 Hardware Setup

### 2.1.1 Selecting Temperature Sensors

The selection of temperature sensors is crucial for accurate data collection. Two types of sensors are used in this project: DS18B20 and DHT22. Each has its own advantages and specific use cases.

**DS18B20 Digital Temperature Sensor**:

* **Accuracy**: ±0.5°C ensures precise temperature readings.
* **Range**: Suitable for temperatures from -55°C to +125°C, covering most practical applications.
* **Communication Protocol**: The 1-Wire protocol allows for multiple sensors to be connected on a single data line, simplifying wiring and reducing the number of GPIO pins required.
* **Advantages**: High accuracy, simple integration, and robustness in various environments.

**DHT22 Temperature and Humidity Sensor**:

* **Accuracy**: Temperature accuracy of ±0.5°C and humidity accuracy of ±2-5% RH.
* **Range**: Temperature range from -40°C to +80°C and humidity range from 0-100% RH.
* **Communication Protocol**: Uses a single digital pin for communication, making it easy to connect to microcontrollers.
* **Advantages**: Provides both temperature and humidity readings, making it versatile for applications requiring both parameters.

These sensors are chosen to provide a comprehensive monitoring solution, with the DS18B20 focusing on precise temperature measurement and the DHT22 offering additional humidity data.

### 2.1.2 Sensor Calibration

Calibration ensures that the temperature readings are accurate and reliable. The process involves:

* **Baseline Measurement**: Measuring the temperature in a controlled environment using a calibrated reference thermometer.
* **Sensor Adjustment**: Adjusting the sensor readings in the code to match the reference thermometer readings.
* **Repeated Tests**: Performing multiple measurements at different temperatures to ensure consistent accuracy across the sensor's range.

### 2.1.3 Connectivity Setup

Establishing reliable connectivity between the sensors and the microcontroller is essential for real-time data transmission. The setup involves:

* **Wiring**: Connecting the DS18B20 and DHT22 sensors to the microcontroller, ensuring proper data, power, and ground connections.
* **Power Supply**: Providing a stable power source to the sensors and microcontroller.
* **Communication Protocols**: Configuring the microcontroller to read data from the sensors using the 1-Wire protocol and then publishing this data to the MQTT broker.

## 2.2 Software Development

### 2.2.1 MQTT Broker Configuration

The MQTT broker is the central component that manages message routing between publishers (sensors) and subscribers (Node-RED). The configuration steps include:

* **Broker Selection**: Choosing an MQTT broker, such as Mosquitto, which is lightweight and open-source.
* **Installation**: Setting up the MQTT broker on a server or a cloud platform.
* **Security Configuration**: Implementing security measures such as SSL/TLS encryption and user authentication to protect data transmission.
* **Topic Structure**: Designing a topic structure for organizing the data, for example, home/temperature/sensor1.

### 2.2.2 Node-RED Flow Design

Node-RED is used to design the data flow, process incoming temperature data, and prepare it for visualization. The design process includes:

* **Node Configuration**: Adding and configuring nodes for MQTT input, data processing, and output.
* **Flow Design**: Creating a flow that subscribes to the MQTT topics, processes the temperature data, and routes it to the dashboard.
* **Error Handling**: Implementing nodes to handle errors and ensure reliable data processing.

### 2.2.3 Dashboard Creation

Node-RED dashboards provide a user-friendly interface for visualizing the temperature data. The creation process includes:

* **Layout Design**: Designing a dashboard layout that includes charts, gauges, and tables to display real-time and historical temperature data.
* **Widget Configuration**: Adding and configuring widgets to display the temperature readings, trends, and alerts.
* **User Interaction**: Implementing interactive elements such as buttons and sliders for user control and customization.

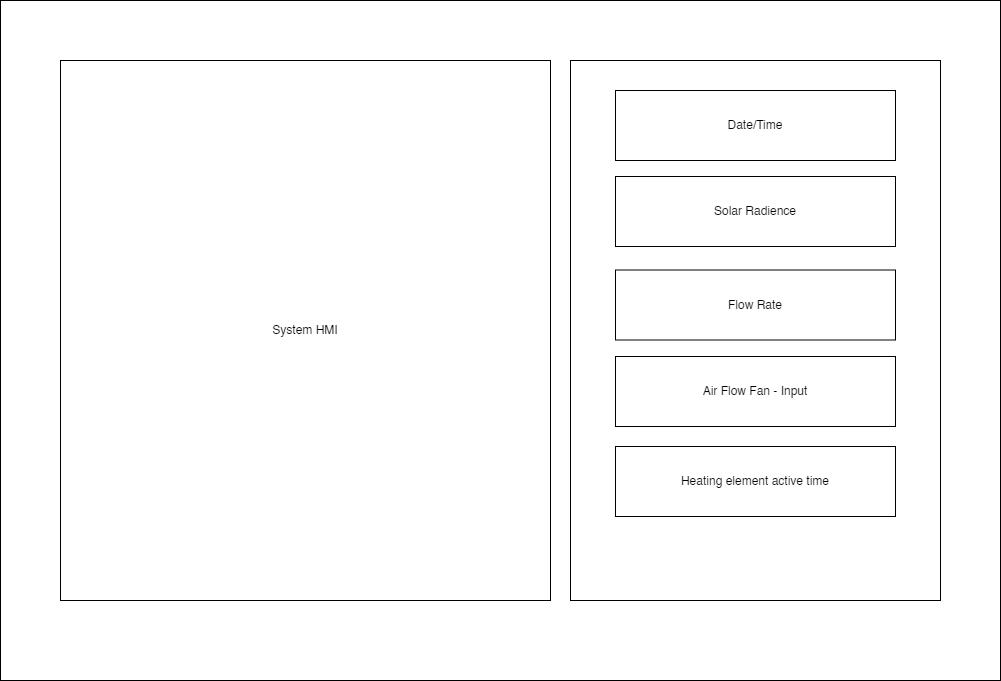
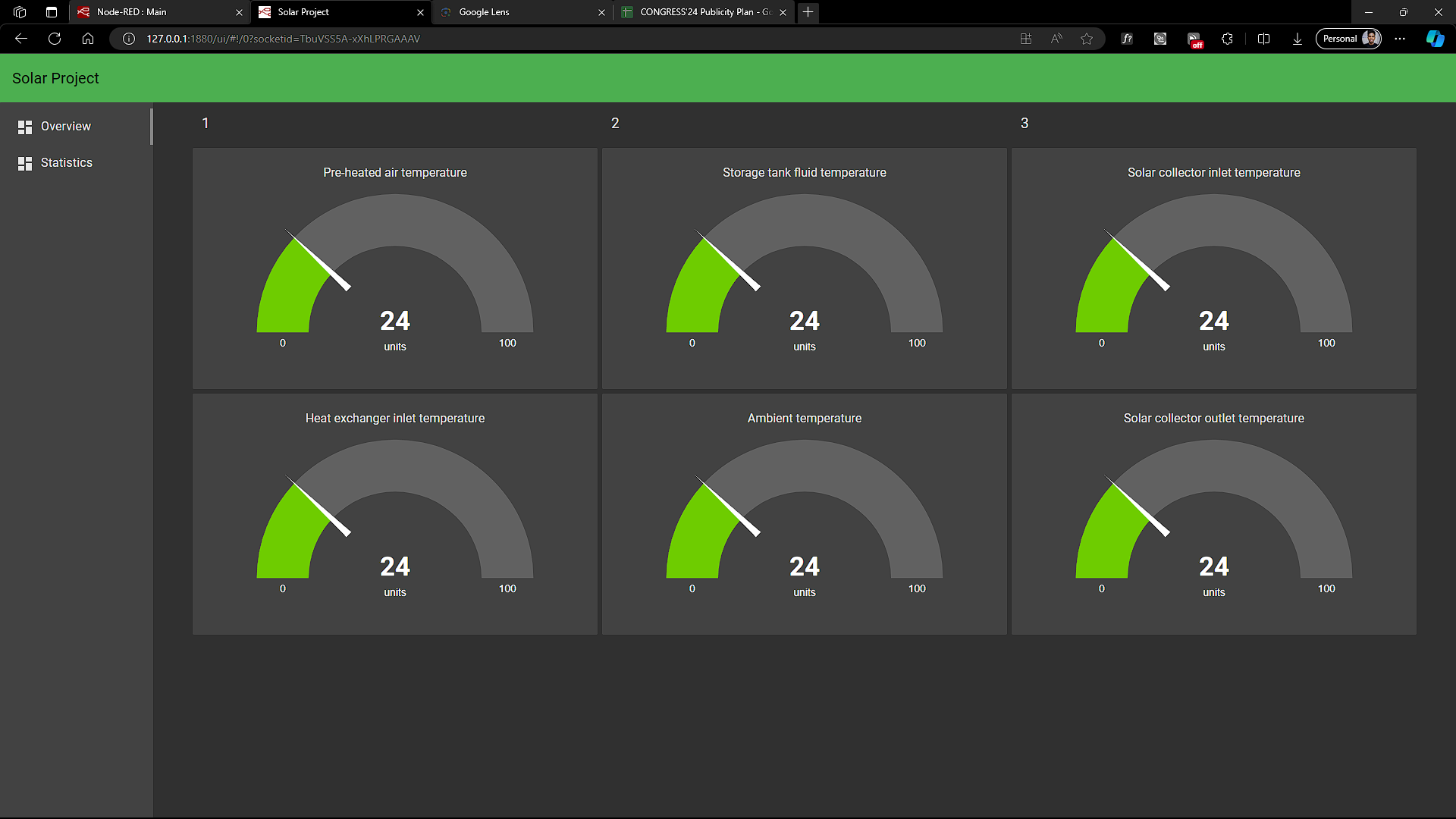
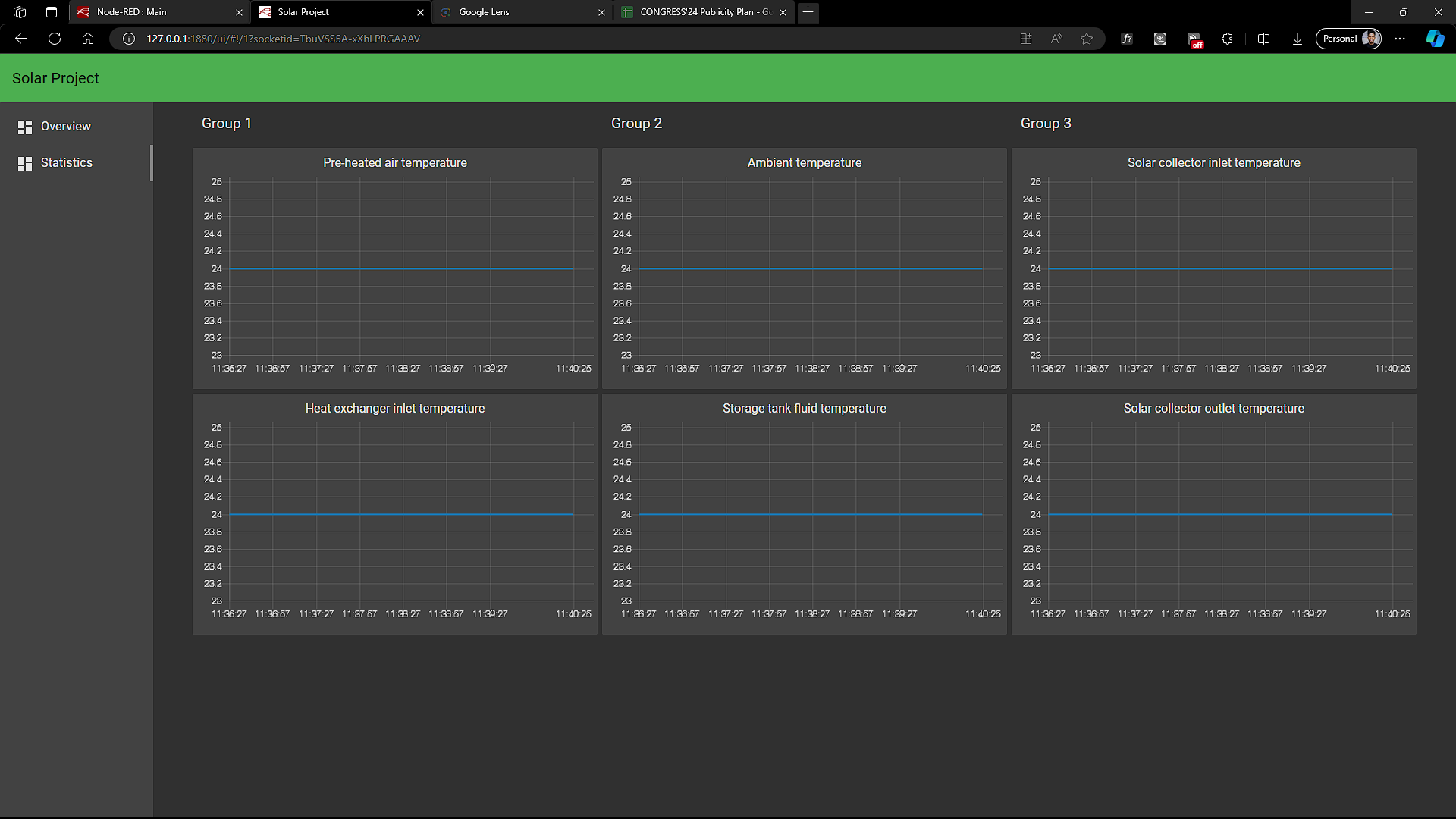


Figure 2. Dashboard layout

## 



## 2.3 Integrating Hardware and Software

The final step is to integrate the hardware components with the software to create a fully functional IoT temperature monitoring system. This involves:

* **Testing Connectivity**: Ensuring that the sensors are correctly connected and communicating with the microcontroller.
* **Data Transmission**: Verifying that the microcontroller is publishing data to the MQTT broker and that Node-RED is subscribing to and processing this data correctly.
* **Dashboard Validation**: Checking that the data is accurately displayed on the Node-RED dashboard and that all widgets are functioning as intended.
* **System Optimization**: Fine-tuning the system for performance, including optimizing data transmission rates, processing times, and dashboard refresh rates.
* **Final Testing**: Conducting comprehensive tests to ensure the system operates reliably under various conditions and scenarios.

# 3. Budget

|  |  |
| --- | --- |
| **Component** | **Model** |
| Microcontroller | ESP32 |
| Thermo sensors | DHT11 |
|  | DS18B20 |
| Flow Meter | YF-S201 |
| Water Pump |  |
| DC – Motor Controller | 1803BK |
| AC – Motor Controller |  |
| Voltage regulator – 5V | LM2596 |
| Power Supply - 12V |  |
| Power Supply – 3V |  |
| Temperature Controller | XH-W3001 |
| Wire(3) |  |
| Soldering Lead |  |
| Pin Headers | Female to male |
| Dot Board | Large |