# Real Time Weather Data Displayment System

*Michael Maloney and Michael Sorce*
Rowan University

December 18, 2018

## 1 Design Overview

The Real-Time Weather Displacement System's main purpose is to represent current weather data in a convenient and coherent manner. The system requires minimal to no user input to operate. The display consists of ten blue LEDs that act as a gauge for weather data, and one RGB LED that indicated what data is currently being displayed on the gauge. The current types of data that can be displayed are temperature(℉), humidity (%), and wind speed (mph). The system is comprised of three major hardware components, an MSP430F5529 development board, a ESP8266 WiFI-Module, and breadboard LED display. Weather data is taken from a weather API by the WiFi-Module (based on desired location), sent to the MSP430 processor over UART, which then deciphers the message and outputs data to the LED display.

### 1.1 Design Features

These are the design features the Real Time Weather Data Display System:

- Data Display
- WiFi Compatibility through ESP8266 Module
- UART Compatibility
- Button Denouncing
- Timer Controlled System
- Closed Loop System (If desired)
- ASCII Decoding and Data Interpreting

## 1.2 Featured Applications

- Convenient Indoor Weather Display of Outside Weather
- Scaled Data Display (0 - 10 magnitudes)

## 1.3 Design Resources

The following is a link to the code and design folders generated during product development. This includes code used to run the MSP430F5529 and ESP8266, as well as additional useful documents related to this product:

GitHub Repository for Weather Data Display System

## 1.4 Block Diagram

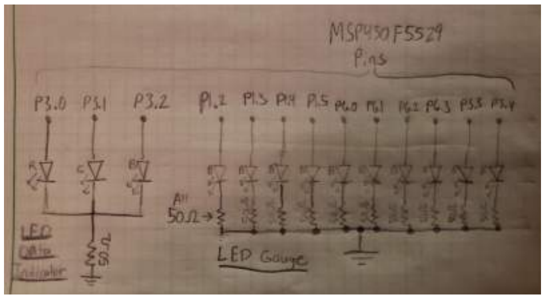

Figure 1: Block Diagram: MSP430F5529 Connections to Breadboard

Figure 1 shows how the MSP430F5529 connects to the LED display built on the breadboard. This shows which MSP pins are connected where, how the LED display is configured, and the overall layout of the circuit design.

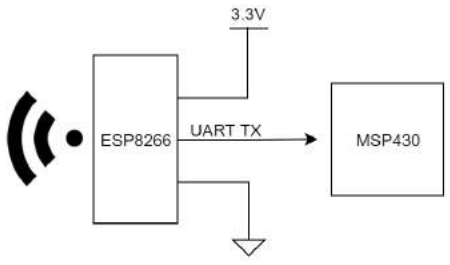

Figure 2: Block Diagram: ESP 8266 Connection to MSP430F5529

Figure 2 shows the block diagram of the ESP8266 wifi module. Since the module is simple, it does not require many connections and only has one connection to the

MSP since UART TX is transmitted through once cable. The only other input to the device is wifi.

### 1.5 Board Image

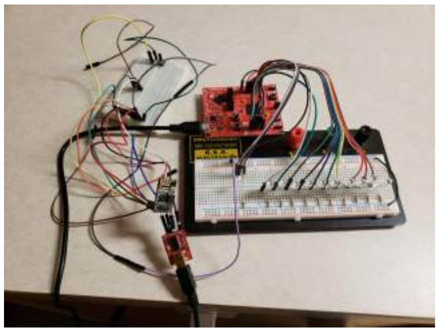

Figure 3: Image of Full Weather Data Display System

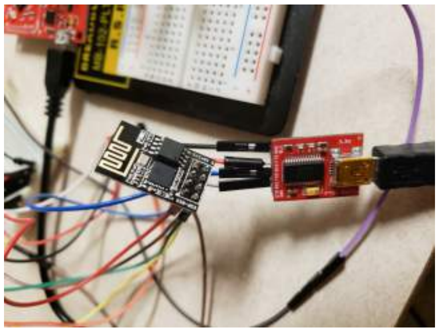

Figure 4: ESP8266 With Connection to Arduino for Coding and Debugging

## 2 Key System Specifications

Table 1: System Specifications

| | |
|---|---|
| Baud | 9600bps |
| Clock Speed | 1 MHz |

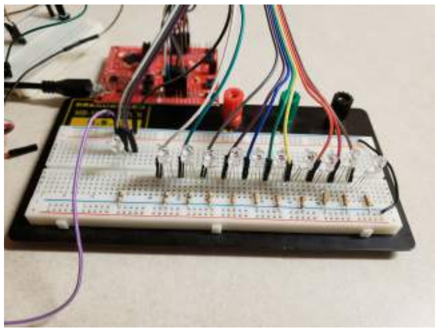

Figure 5: Image of LED Gauge Circuit on Breadboard

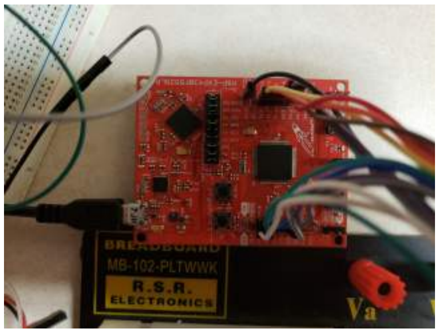

Figure 6: MSP430F5529 with Pins connected to Breadboard

## 3 System Description

Since living in Glassboro it has been a hassle dealing with unexpected weather conditions. This is why we decided to create an LED guage that will display weather information acquired from a wifi module that can connect to the internet. The wifi module will send this data to the MSP board, which will drive the LED display. This will serve as a quick and easy way of seeing the temperature in the morning without going through a weather service.

The system starts with the ESP8266 wifi module that gets the weather information from an online API. This information is parsed in the ESP8266 and sent to the MSP430F5529 over UART. The information is then displayed on LEDs from the MSP430 and toggled with a button press.

### 3.1 Highlighted Devices

- MSP430F5529
  - This launchpad receives data from the ESP8266 and displays data on LEDs.

- ESP8266
  - This module collects data from wifi and sends it to the MSP.

### 3.2 Device/IC 1

The MSP430F5529 is used to receive weather data through UART from the wifi module and display that data on LEDs. Since only 10 LEDs are used for the display, and 3 information types are to be shown, another RGB LED is set up. The 10 LEDs are used as a type of thermostat, where 1 LED on means a low temperature, wind speed, or humidity, and 10 LEDs on means the highest for all (in the range we selected). The values in between are broken up in steps of 10. The RGB LED is used to decipher which dataset is shown. The RGB value is pin driven and turns red when temperature is displayed, green when humidity is displayed, and blue when wind speed is displayed.

### 3.3 Device/IC 2

The ESP8266 module was used as a connection to the internet to receive a constantly updating string of weather information. The module would connect to the OpenWeatherMaps API and parse the Glassboro data for wind speed, temperature, and humidity. This data would then be packaged with a specific letter before each variable. The letter would clarify which variable is which so the MSP can decode the values properly.

## 4 SYSTEM DESIGN THEORY

The two main components of the system are the ESP8266 wifi module and the MSP430F5529 launchpad. Both have their very specific used and could not function without one another.

The system starts with the ESP8266 wifi module. The module connects to the internet and collects weather data. This data is modified in the system and then sent to the MSP430F5529

The MSP430F5529 receives the data over UART and parses it. One variable of the data (either temperature, wind speed, or humidity) is then displayed on a string of 10 LEDs in a sort of thermometer-like fashion. The other data variables can then be cycled through by a button press.

### 4.1 Design Requirement 1

The ESP8266 module connects to the internet via a client-server connection. This then connects to the OpenWeatherMaps weather API. Many different types of data could have been used, but current temperature, wind speed, and humidity were selected. The whole string of data on the API is transferred to the wifi module. The data is parsed with the use of JSON, and the select weather values are then set to variables. These values are packaged so that the MSP board can effectively received the data without error and (much) noise. Before each value is a letter corresponding to the value type. An example of the package would look like: T35H09S12. The T correlates to temperature, H correlates to humidity, and S correlates to wind speed. If any value is less than 10, a 0 is inserted before the value to make it two digits, seen as ”H09”. This values would then be sent over the TX UART pin to the MSP.

### 4.2 Design Requirement 2

The MSP430F5529 microcontroller was selected because of its familiarity to the group. Instead of spending time and resources on understanding another board, this board was chosen.

In the system, the MSP430F5529 acts as the driving circuitry to all LEDs. Each of the 10 LEDs in the the ”thermostat” is connected directly to an I/O pin due to the simplicity of the design. Since the board has so many pins, it seemed impractical to wire up any other configuration given the time requirements. The final 11th LED, the RGB LED, was connected via three I/O pins, instead of PWM. The RGB LED acts as three LEDs of the aforementioned red, green, and blue colors. Since only three weather values are displayed, it is unnecessary to PWM the inputs. Switching between weather values is done through a button press. Since bouncing is an issue with MSP buttons, software debouncing was incorporated into the code.

The ESP8266 sends the data as ASCII values to the MSP. The MSP board collects these values and parses them to see if the correct package is obtained based on the predefined package setup (i.e. letters other than T, H, and S are ignored). This both cuts noise and increases accuracy. The values in the code are then converted from ASCII and set to a variable. This variable is then converted to a number from 0 to 10 because of the LED setup. All values are out of 0-100, so a 45 percent humidity would display 5 LEDs, since 5 would round upwards (44 would show 4 LEDs).

## 5 Getting Started/How to use the device

When first getting this device, it must be configured. The code needs to be altered to receive weather data for the desired town. This is a temporary set up for the prototype model. For a final marketable model, there would be a minor user interface where a user can enter a zip code, and it will adjust the code accordingly.

After the device is configured to the right location, the WiFi-Module needs to establish an internet connection. This must be done through the code as of now, but would be altered for a future model. Once a connection is selected for the first time, the WiFi-Module will automatically connect to the same WiFi whenever it is powered on.

Once the WiFi module is configured fully (and all devices are powered by some means i.e. outlet/batteries) the device is ready to function. Data will be automatically received from the API and sent over UART to the processor every 10 seconds. The

MSP430F5529 processor will interpret update the LED display accordingly.

Currently, the only part of this device that requires user interaction after set up is the button P1.1 on the MSP430F5529, which is used to cycle through which data is displayed. For a final design, this will be taken out and the data will cycles on an infinite loop with delays at each step.

For the user to interpret the data that is being displayed, they need to reference the key that comes with the device. This will tell the user which data set is being displayed based on the color of the RGB LED indicator, and what the 0-10 LED scale represents in understandable values (i.e. degrees or %).

## 6 Getting Started Software/Firmware

Ideally the ESP8266 wifi module should already be setup, but this can sometimes be an issue. There is no user interface with the code at this point, but reprogramming the device through the serial monitor and Arduino programming can fix this.

Because wifi can sometimes be an issue, debugging the ESP device while it is plugged into a laptop and the MSP board is ideal. Debugging prompts are featured in the provided Arduino code. Changing the wifi SSID and password is possible through the code. Everything sent over UART to the MSP is also visible in the Arduino serial monitor. This can help finding discrepancies between what is seen in the MSP and what is sent from the ESP.

As previously stated, the ESP8266 should work on its own without any software involvement. Based on wifi and location differences, it is possible to change these two variables if wanted in the Arduino code.

## 7 Test Setup

To setup this device for testing, the ESP6288 needs to be connected to a WiFi source, and the MSP and ESP need to be powered by some means. Once this is done, the device should update autonomously. To check if the data is accurately being represented, one can check the current weather values online, or use a serial communication program to read what values are being transmitted by the MSP. This is because all values received by the MSP are echoed for debugging purposes.

### 7.1 Bill of Materials

- MSP430F5529
- ESP8266
- Jumper Wires
- 10 LEDs + Resistors

- 1 RGB LED + Resistors