Application Note

25 December 2017

WiFi updating queue counter using MSP430G2553, LV-MaxSonar-EZ1 and ESP8266

Stephen Glass
Electrical & Computer Engineering, Rowan University
Relevant Github Page: https://github.com/glasss6/msp430-queueCounter

1. Abstract

An embedded system to detect the length of a queue is developed using the MSP430G2553 microprocessor, LV-MaxSonar-EZ1 distance sensor and ESP8266 WiFi module. Queue length and historical statistics are visualized on a live-updating webpage.

2. Introduction

Food establishments in highly concentrated areas such as Rowan University experience long lines during peak times (e.g transitioning class periods). Consumers in the area may wonder the best time to visit the establishment to wait on the shortest line possible. Queue Counter can be developed to detect the amount of people waiting in a line. The current length of the line can be viewed real-time on a live-updating website.

The system uses an Ultrasonic distance sensor (LV-MaxSonar-EZ1) to determine the distance between the system and the destination. Figure 1 below demonstrates the usage of the distance sensor to calculate the total length of the queue.

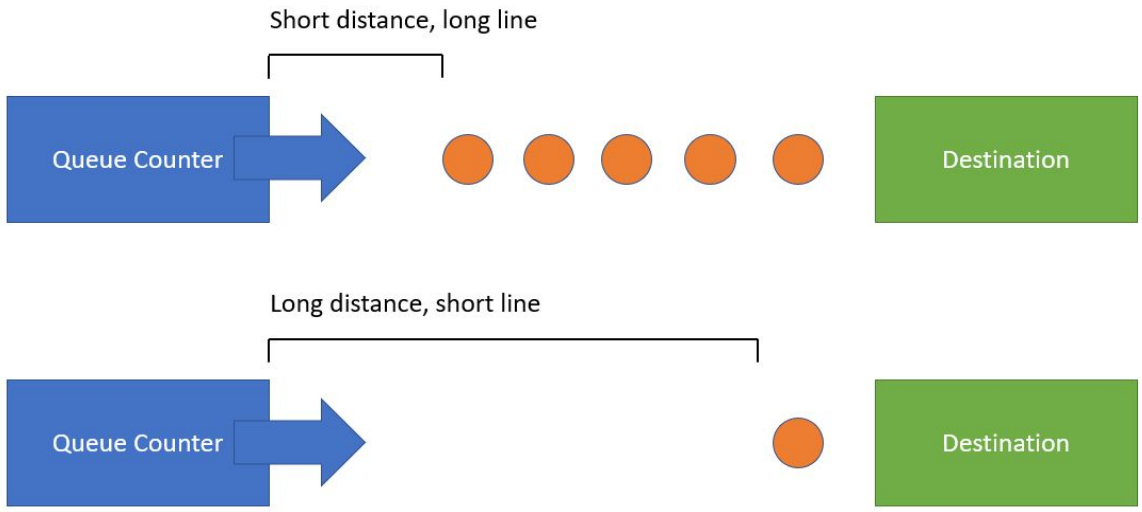


Fig. 1: Relationship of distance to queue length

The MSP430G2553 program interprets the distance from the LV-MaxSonar-EZ1 and formats the data to the ESP8266 WiFi module for upload to the web server. The web server handles the requests and inserts the information into a database. The visualization of the database can be seen in Figure 2 below.

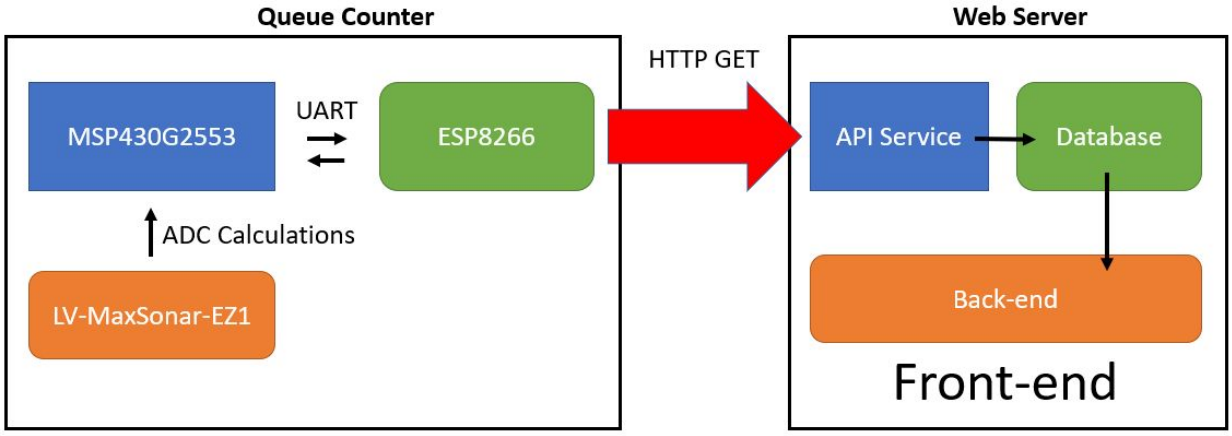


Fig. 2: Visualization of the queue counter data path

3. Background

3.1 MSP430G2553

Texas Instruments MSP430G2553 is part of the MSP430 family of ultra-low-power microcontrollers featuring different sets of peripherals. The MSP430G2553 mixed signal microcontroller features a 16-bit RISC CPU, 16-bit registers, 16KB non-volatile memory, 512 bytes RAM, 10-bit ADC, UART, and two 16-bit timers. The G2x53 series is popular due to the low price and various package sizes. The MSP430G2553 acts as the master node (central hub) of the queue counter – connecting the distance sensor and ESP8266.

3.2 ESP8266

ESP8266 WiFi module is a self-contained SoC with integrated TCP/IP protocol. The module is an extremely low-cost all-in-one solution for WiFi connectivity for microcontrollers. The ESP8266 contains an API (AT command set firmware) which allows any microcontroller to control the module using UART protocol. The ESP8266 is used in the queue counter as a client to initiate TCP connections to a web server and transmit information via HTTP requests.

3.3 LV-MaxSonar-EZ1

LV-MaxSonar-EZ1 provides short to long-range detection and ranging in a small package. The sensor operates using 42KHz ultrasonic waves for object detection. The module is able to interface with Serial, Analog voltage, or Pulse width. Queue counter interfaces with the LV-MaxSonar-EZ1 using Analog voltage to ADC input channel. The sensor was chosen due to the simplicity of output interfaces.

4. System Design

4.1 Schematic and PCB

The queue counter schematic was designed using DipTrace and can be seen in Figure 3 below.. The circuit interfaces the MSP430 microprocessor with both the ESP8266 and LV-MaxSonar-EZ1 distance sensor. The circuit provides several second-order design considerations such as bulk, bypass capacitors, power regulation, and LED status indicators.

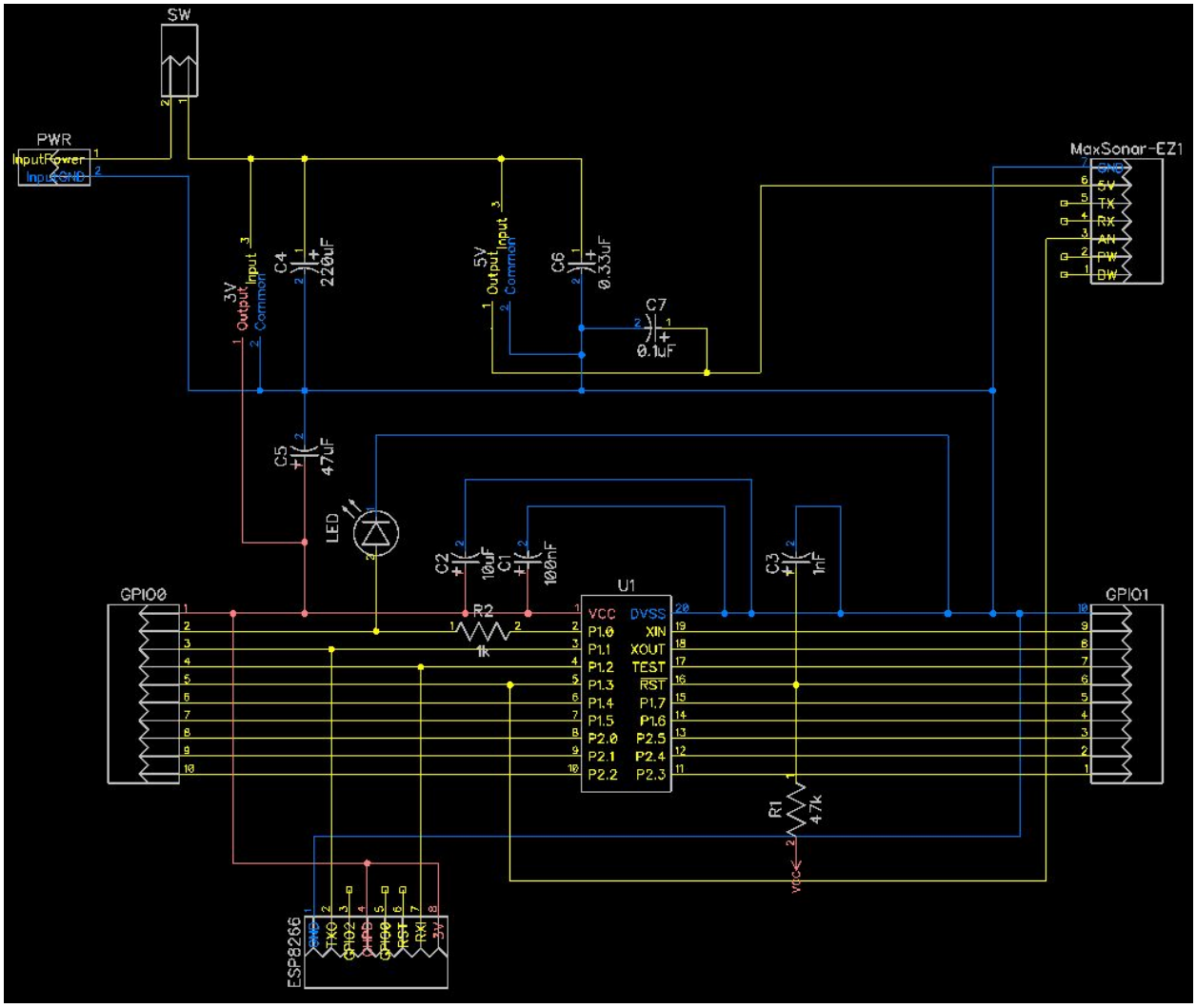


Fig. 3: Schematic of the queue counter with interfaces to ESP8266 and LV-MaxSonar-EZ1

The LV-MaxSonar-EZ1 requires 5V power for max range. However, the MSP430G2553 and ESP8266 require 3.3V. Therefore, two voltage regulators are used for power isolution in the system. The system main power is designed to be powered from 6V. This is equivalent to 4xAA batteries.

A PCB was designed for the queue counter using DipTrace and manufactured by PCBWay. A through-hole PCB was designed due to easy solderability and readily available components. The physical dimensions of the PCB should not be larger than a typical 4xAA battery pack. A 3D visualized render of the PCB is seen in Figure 4 below.

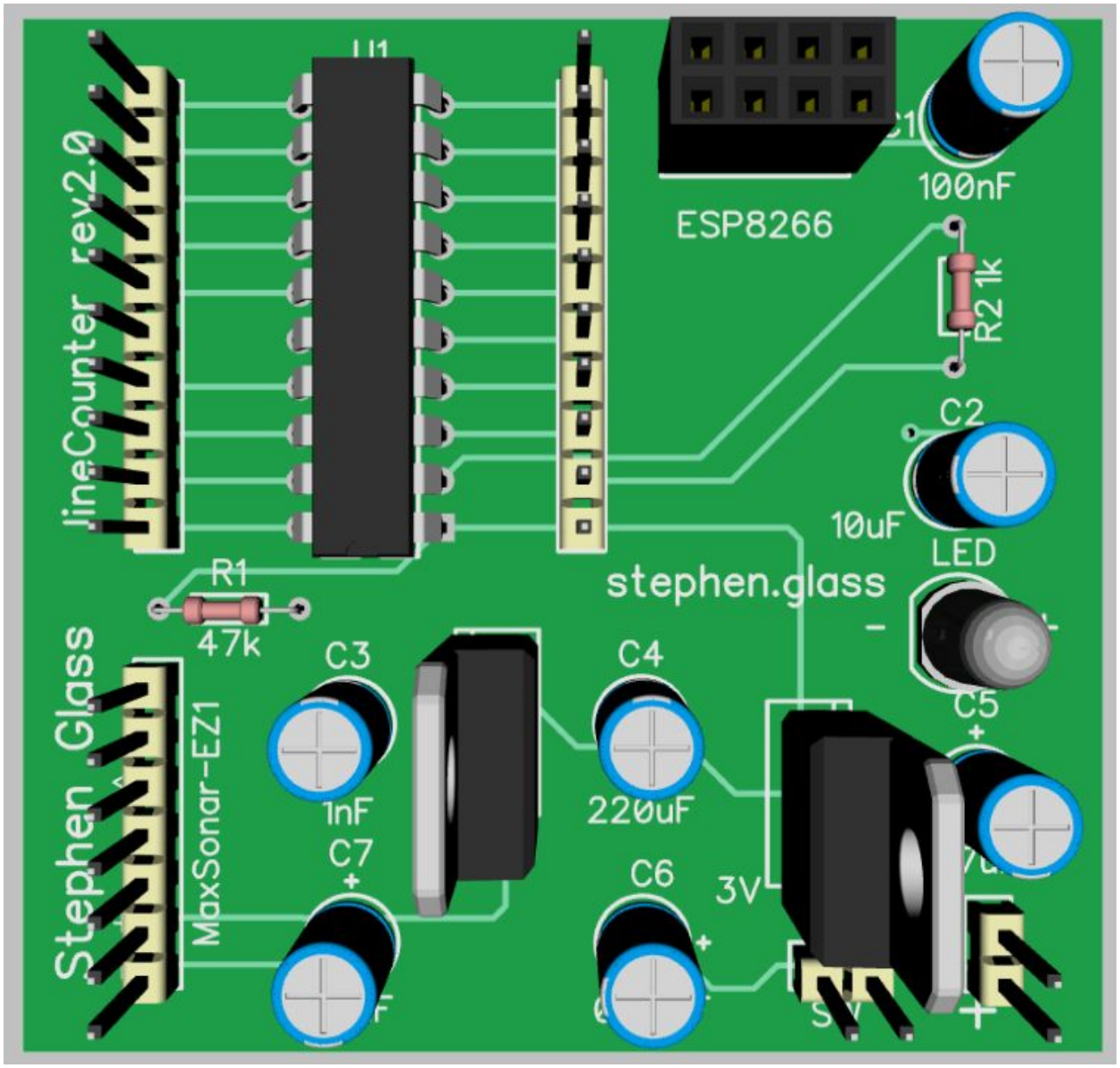


Fig. 4: 3D render of the queue counter printed-circuit board

4.2 MSP430G2553 Software

4.2.1 ADC Initialization

Analog-to-digital conversion is used to convert the analog voltage output from the LV-MaxSonar-EZ1 sensor to a readable distance in inches. LV-MaxSonar-EZ1 outputs the distance as (Vcc/512) / inch.

With Vcc = 5V the sensor outputs 9.8mV per inch. The sensor has a max distance range of 254 inches. Therefore, the maximum voltage the sensor can output is 2.49V. This works out well because the MSP430G2553 has a voltage reference setting for 2.50V which will allow us to read the full range of results. Note, according to the MSP430G2553 datasheet 2.5V reference is only available on P1.4 (V_{REF+}).

Using a 10-bit ADC, a voltage reference setting of 2.5V, and sensor output of 9.8mV per inch we can develop the following equation to convert the input to a readable distance.

$$\text{Distance} = \text{ADC Input} \times (\frac{2.5V}{1024} \times \frac{1\text{inch}}{9.8mV})$$

The ADC ISR solves the distance equation using the ADC input. The MSP430 ADC registers are configured to the following parameters.

ADC Register Initializations

Register	Bit	Description
ADC10CTL0	ADC10SHT_1	ADC Sample-and-Hold time: 8 x ADC10CLKs
ADC10CTL0	SREF_1	Select reference: VR+ = VREF+ and VR- = VSS
ADC10CTL0	REFON	Reference generator: ON
ADC10CTL0	REF2_5V	Reference generator voltage: 2.5V
ADC10CTL0	ADC10ON	ADC10 Enable: ON
ADC10CTL0	ADC10IE	ADC10 Interrupt: ENABLED
ADC10CTL1	INCH_4	Input channel A4 (P1.4)
ADC10AE0	BIT4	ADC Analog enable A4

4.2.2 UART Initialization

UART is used to communicate (RX/TX) with the ESP8266 WiFi module. The ESP8266 communicates at 115200 baud with 3.3V logic.

UART Register Initializations

Register	Bit	Description
DCOCTL	0	Select lowest DCOx and MODx settings
DCOCTL	CALDCO_1MHZ	Set DCO
BCSCTL1	CALBC1_1MHZ	Set DCO
P1SEL	BIT1 + BIT2	P1.1 = RXD, P1.2=TXD
P1SEL2	BIT1 + BIT2	P1.1 = RXD, P1.2=TXD
UCA0CTL1	UCSSEL_2	Set UART use SMCLK
UCA0BR0	8	1MHz 115200 baud
UCA0BR1	0	1MHz 115200 baud
UCA0MCTL	UCBRS2	Modulation UCBRSx = 5
UCA0MCTL	UCBRS0	Modulation UCBRSx = 5
UCA0CTL1	~UCSWRST	Initialize USCI state machine

Functions are created to send and receive UART strings to/from the ESP8266.

Function	Description
`void TX(const char *s)`	Send a string via UART
`void putc(const unsigned c)`	Send a character via UART
`void crlf(void)`	Sends a carriage return and new line via UART
`int calcLen(const char *s)`	Returns the length of a string
`int readUART(char *search, char len, int delayParam);`	Reads incoming UART for a specified string. Time out after a certain delay period.

4.2.3 Timer Initialization

A timer is initialized to send distance data to the web server on a specified interval. The MSP430G2553 contains a 1MHz SMCLK. The maximum clock divider available is 8-times. Because the MSP430G2553 uses 16-bit registers the maximum period is 65535. Therefore, the slowest hardware frequency for a timer is 2Hz.

$$2\text{Hz} = \frac{(1\text{MHz})/8}{\text{Period}}$$

Solving for the timer period for a 2Hz interval results in a period of 62500. This is the maximum value before we reach an overflow of 65535.

Timer A Register Initializations

Register	Bit	Description
TA0CCTL0	CCIE	Enable interrupts for Timer A0
TA0CTL	TASSEL_2	Select SMCLK clock source
TA0CTL	MC_1	Set timer mode to UPMODE
TA0CTL	ID_3	Set clock divider to SMCLK/8
TA0CCR0	62500	Set capture compare register period to 62500

Using software, the timer can be acted upon in slower intervals. Because we know the timer interrupts every 2Hz we can use software to determine when to activate code. Definitions can be set to determine the rate at which data is uploaded to the server and how often the distance sensor is sampled. The program by default uploads data to the server every 10 seconds and samples the sensor every 5 seconds.

Timer Constant Definitions

Constant	Description
UPLOAD_RATE	Rate at which sensor data is uploaded to web server [0.5 * s]
SAMPLE_RATE	Rate at which sensor data is sampled for ADC conversion [0.5 * s]

4.2.4 WiFi Functions

The program has functions for sending specific commands to the ESP8266 to send and format data for upload.

The program requires definitions for WiFi information such as SSID, Password, URL, etc.

WiFi Constant Definitions

Constant	Description
MAX_SEND_LENGTH	Determines maximum number of digits of sensor data to send (default 3)
BASE_URL	The first part of the URL for API request servicer
BASE_URL_END	The second part of the URL for API request servicer
TCP_REQUEST	Preformatted message to initiate TCP request
WIFI_NETWORK_SSID	SSID for WiFi network
WIFI_NETWORK_PASS	Password for WiFi network
BASE_URL_LEN	Length of BASE_URL
BASE_URL_END_LEN	Length of BASE_URL_END
ENABLE_CONFIG_WIFI	Enable to configure the WiFi network to settings

Functions are created perform WiFi related tasks.

Function	Description
`void WifiConfigureNetwork(void)`	Configure the network to defined settings
`void WiFiSendTCP()`	Initiate TCP request with web server
`void WiFiSendLength(unsigned int data)`	Send the length of the incoming TCP request to ESP8266 before upload (Requires processing to convert integer to ASCII character digits)
`void WiFiSendMessage(unsigned int data)`	Send the message with data to web server for upload

The program uses the ESP8266 AT command set to perform actions on the ESP8266 over UART. For example, “AT+CWMODE_DEF=1” over UART sets the ESP8266 to client-mode.

For the ESP8266 to send data to the server it must first initiate the TCP connection, receive the length of the message to send to the server, receive the message, then send the message to the API service on the server as an HTTP GET request. This process is detailed in Figure 5 below.

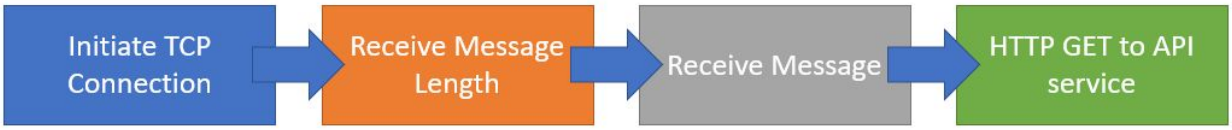


Fig. 5: Process for sending data using the ESP8266

4.2.5 Low Power

The program enters low power mode in between sampling and upload periods. The MSP430G2553 enters low-power-mode 0 which shuts off the CPU and enables interrupts. The ESP8266 is set to modem-sleep which will idle the WiFi modem when not in use. The ESP8266 will use about 15mA on average during this mode.

4.3 Web Back-end Software

4.3.1 API Servicer

Referring back to Figure 1, the queue counter sends data from the ESP8266 to the API servicer. The API service software inserts the data into a database. The API service is PHP software hosted on the web server. The code looks for a HTTP GET request where the HTTP input is the distance from the LV-MaxSonar-EZ1 sensor.

The API updates a MySQL database for the following parameters:

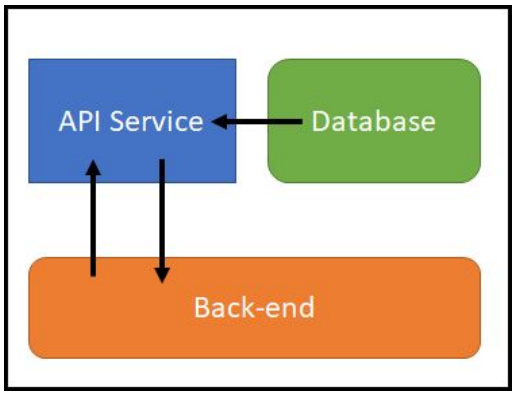
- Current distance
- Average distance
- Minimum distance
- Maximum distance
- Last updated time

4.3.2 Back-end Web Code

The back-end web code is the code that fetches distance information from the database and formats data to be displayed on a real-time updating webpage. The back-end code is written in JavaScript and calls PHP API requests for JSON encoded database information.

The back-end code has timers which call an API service on specified intervals to fetch information from the database for distance information. The back-end JavaScript code calls a JSON API request to an API service. The API service receives information from the database, encodes the information in JSON

Fig. 6: Back-end data retrieval



and sends it back to the initiating service. The process is visualized in Figure 6 above.
The JavaScript back-end code is programmed to update HTML elements with the distance data and statistics of the queue.

4.4 Website Visualization

Queue counter data is visualized on a real-time updating website. The webpage displays the estimated number of people in line and the length of the line. Additionally, it also provides statistics for average, minimum, and maximum queue lengths. A screenshot of the webpage is seen below in Figure 7.

Fig. 7: Screenshot of the real-time webpage to view queue information

The webpage queries the database every 10 seconds for updated information. The webpage will automatically update the content to reflect the updated information. A user could view the website for queue information before visiting a store.

4.5 Bill of Materials

Part #	#	Purpose
MSP430G2553	1	Microprocessor hub of system
ESP8266	1	WiFi module
LV-MaxSonar-EZ1	1	Distance sensor to detect people
1nF capacitor	1	Timing for MSP430 programming
100nF capacitor	1	Bulk/bypass capacitor
0.1uF capacitor	1	Bulk/bypass capacitor
0.33uF capacitor	1	Bulk/bypass capacitor
10uF capacitor	1	Bulk/bypass capacitor
47uF capacitor	1	Bulk/bypass capacitor
220uF capacitor	1	Bulk/bypass capacitor
47k Ohm resistor	1	Reset pull-up resistor
1k Ohm resistor	1	LED current limiting resistor
LED	1	LED for status indicating
UA78M33C	1	3.3V Linear switching voltage regulator
L7805	1	5.0V Linear switching voltage regulator
SPST/SPDT Switch	1	Power switch
1 x 10 male headers	2	Headers for MSP430 pins
2 x 4 female headers	1	Headers for ESP8266 input
1 x 7 male headers	1	Headers for LV-MaxSonar-EZ1 input

5. Summary of System Performance

The system successfully accomplishes the goal of measuring the length of a queue. A user is able to view a website to observe the length of the queue at any time. For future improvement, a longer range distance sensor could be used or a different method for people detection such as image recognition / computer vision. Additionally, the ESP8266 WiFi module could be put into a deeper-sleep mode when not uploading data to conserve more battery.

6. Appendix

Picture of the assembled PCB queue counter connected to 4xAA batteries for power.