Wearable & Ubiquitous Computing

CSCE 4114

Sprint 2019

Mon, Wed, Fri (12:55 PM – 1:45 PM)

Assignment 3

Context-Aware To-Do Badge

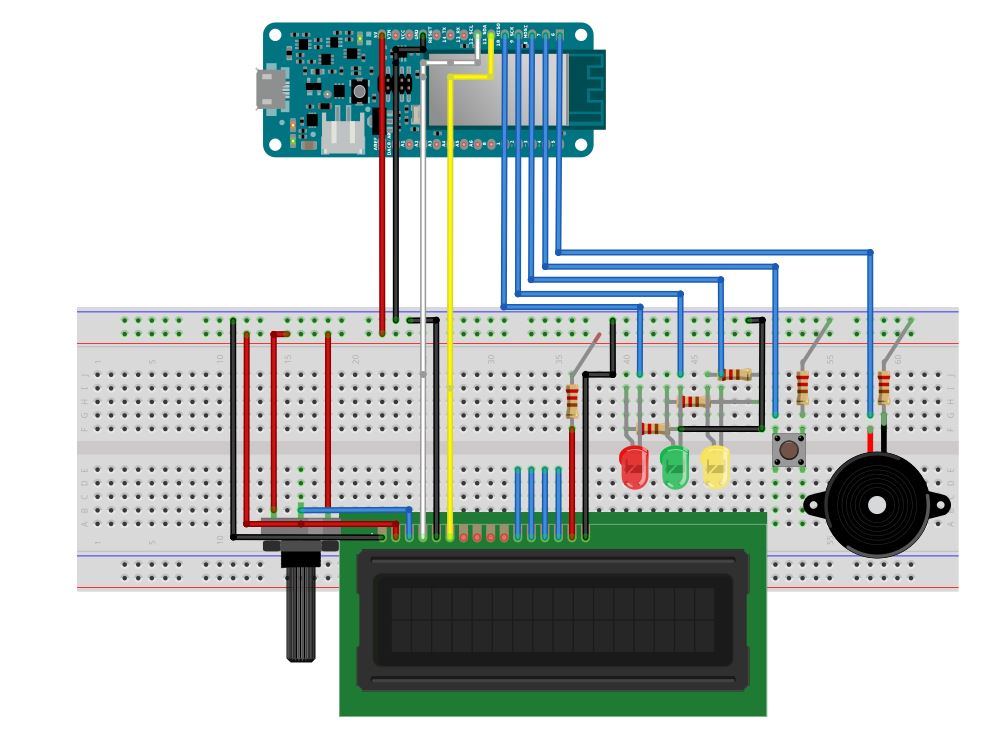
4-19-2019

Suhail Basalama

010786839

[sebasala@email.uark.edu](mailto:sebasala@email.uark.edu)

# Diagram



# Introduction

In this project, I created a system that displays to-do tasks for the user. The system is self-explanatory and user-friendly where the user can see lights changing as the deadline for the earliest task approaches. A piezo element makes a tone once the deadline passes. An LCD screen is used to display the to-do items, and the user can scroll the list using a push button. The push button is also used to remove a task if it passed the due date. The system is setup as shown in the diagram above.

# Implementation

If there is a WIFI connection, the system tries to connect to two servers: MQTT server for the incoming messages, and an NTP server to get the current time. Messages coming from the MQTT are JSON lists, and they are deserialized into to\_do\_list objects using the JsonArduino library version 6.

The libraries used in the program are all from the Arduino library manager as follows:

#include <FlashAsEEPROM.h> //FlashStorage

#include <FlashStorage.h>

#include <ArduinoJson.h> //ArduinoJson

#include <WiFi101.h>

#include <NTPClient.h>

#include <WiFiUdp.h>

#include <time.h>

#include <PubSubClient.h>

#include <LiquidCrystal.h>

On boot up the system connects to WIFI, MQTT server, NTP server, update the current time, and gets the most recent list from flash memory, otherwise initialize the list to very large time and NONE name for each item. In the main loop, the system checks for WIFI connection every 10 seconds and for new messages from the MQTT server every second. Every 60 seconds, the system updates the time from the NTP server. If the system is not online, the current\_time is set to zero. After that the logic for the scrolling and states of the system are implemented.

When a new JSON list arrives the system automatically update the list, sort the list by closest deadline, and store the list in the flash memory. Insertion sort is used for sorting the list items by due time.

The time format is as follows:

mm/dd/y hh:mm:ss

# **Results**

This project was much harder than the previous two assignments both in terms of hardware and software setup. I faced many challenges with the libraries and implementation, but I overcome them through unit testing and functions. The system now passes all the 10 cases, and it produces the expected outputs.

## **Source Code**

//Author Suhail Basalama

#include <FlashAsEEPROM.h>

#include <FlashStorage.h>

#include <ArduinoJson.h>

#include <WiFi101.h>

#include <NTPClient.h>

#include <WiFiUdp.h>

#include <time.h>

#include <PubSubClient.h>

#include <LiquidCrystal.h>

#define LEAP\_YEAR(Y) ( (Y>0) && !(Y%4) && ( (Y%100) || !(Y%400) ) )

#define SERVO 7

#define USERNAME "sebasala"

#define TODOLIST\_TOPIC "uark/csce5013/sebasala/todolist"

**typedef** struct **{**

char title**[**36**];**

uint32\_t due**;**

uint16\_t yellow**;**

uint16\_t red**;**

**}** toDoItem**;**

**typedef** struct **{**

toDoItem item\_list**[**5**];**

uint8\_t list\_length**;**

**}** toDoList**;**

//////////////hardware setup//////////////////////

//LCD

const int rs **=** 12**,** en **=** 11**,** d4 **=** 5**,** d5 **=** 4**,** d6 **=** 3**,** d7 **=** 2**;**

LiquidCrystal lcd**(**rs**,** en**,** d4**,** d5**,** d6**,** d7**);**

//Button

const int button **=** 7**;**

//Piezo

const int piezo **=** 8**;**

//LEDs

const int red **=** 10**,** green **=** 9**,** yellow **=** 8**;**

//////////////////////////////////////////////////

///////////////Network setup//////////////////////

//WIFI

char ssid**[]** **=** "Suhail Basalama"**;**

char password**[]** **=** "12345678"**;**

int status **=** WL\_IDLE\_STATUS**;**

//WIFI client

WiFiClient wifiClient**;**

// Define NTP Client to get time

WiFiUDP ntpUDP**;**

NTPClient timeClient**(**ntpUDP**);**

//MTTQ client

const char**\*** mqtt\_server **=** "thor.csce.uark.edu"**;**

PubSubClient client**(**wifiClient**);**

long lastMsg **=** 0**;**

char msg**[**50**];**

int value **=** 0**;**

long lastCheckConnected **=** **-**1000**;**

//////////////////////////////////////////////////

////////////////data structures setup/////////////

//to do list struct

toDoList tdList**;**

char jsonToDoList**[**500**]** **=** ""**;**

//json

// allocate the memory for the document

DynamicJsonDocument doc**(**1024**);**

// FlashStorage Library declaring a variable called my\_flash\_store that is of type toDoList

FlashStorage**(**flash**,** toDoList**);**

//////////////functions///////////////////////////

String getFormattedDate**(**unsigned long t**,** unsigned long secs**=**0**);**

void insertionSort**(**toDoList**\*** tdList**,** unsigned char n**=**5**);**

void printToDoList**(**toDoList**\*** tdList**);**

///////////global helper variables////////////////

unsigned char scroll**;**

unsigned char scroll\_counter**;**

unsigned char scrolling **=** 0**;**

unsigned char wifi\_counter**;**

unsigned char time\_update**;**

uint32\_t current\_time **=** 0**;**

unsigned char passedDue **=** 0**;**

int wifi\_wait**=**0**;**

void setup**()** **{**

pinMode**(**button**,** INPUT**);**

pinMode**(**green**,** OUTPUT**);**

pinMode**(**yellow**,** OUTPUT**);**

pinMode**(**red**,** OUTPUT**);**

Serial**.**begin**(**9600**);**

lcd**.**begin**(**16**,** 2**);**

**while(!**Serial**);**

Serial**.**print**(**"----Application Started----\n"**);**

// set up the LCD's number of columns and rows:

setup\_wifi**();**

client**.**setServer**(**mqtt\_server**,** 1883**);**

client**.**setCallback**(**callback**);**

timeClient**.**begin**();**

timeClient**.**setTimeOffset**(-**5**\***3600**);** //-5 hours GMT

//read to do list from flash

tdList **=** flash**.**read**();**

**if(**tdList**.**item\_list**[**0**].**due **==** 0**)**

**{**

tdList**.**list\_length **=** 5**;**

**for(**int i**=**0**;** i**<**tdList**.**list\_length**;** i**++)**

**{**

strcpy**(**tdList**.**item\_list**[**i**].**title**,**"NONE"**);**

tdList**.**item\_list**[**i**].**due **=** 2000000000**;**

tdList**.**item\_list**[**i**].**yellow **=** 0**;**

tdList**.**item\_list**[**i**].**red **=** 0**;**

**}**

**}**

unsigned char success **=** timeClient**.**forceUpdate**();**

current\_time **=** success**?** timeClient**.**getEpochTime**():**0**;**

**}**

void printToDoList**(**toDoList**\*** tdList**){**

Serial**.**println**(**"Start printing"**);**

**for(**int i**=**0**;** i**<**tdList**->**list\_length**;** i**++)**

**{**

Serial**.**print**(**tdList**->**item\_list**[**i**].**title**);**

Serial**.**print**(**" "**);**

Serial**.**print**(**tdList**->**item\_list**[**i**].**due**);**

Serial**.**print**(**" "**);**

Serial**.**print**(**tdList**->**item\_list**[**i**].**yellow**);**

Serial**.**print**(**" "**);**

Serial**.**print**(**tdList**->**item\_list**[**i**].**red**);**

Serial**.**print**(**"\n"**);**

**}**

**}**

void CopyToDoItems**(**toDoItem**\*** destination**,** toDoItem**\*** source**){**

strcpy**(**destination**->**title**,**source**->**title**);**

destination**->**due **=** source**->**due**;**

destination**->**yellow **=** source**->**yellow**;**

destination**->**red **=** source**->**red**;**

**}**

void insertionSort**(**toDoList**\*** tdList**,** unsigned char n**)** **{**

int i**,** j**;**

toDoItem key**;**

**for** **(**i **=** 1**;** i **<** n**;** i**++)** **{**

CopyToDoItems**(&**key**,** **&**tdList**->**item\_list**[**i**]);**

j **=** i **-** 1**;**

**while** **(**j **>=** 0 **&&** tdList**->**item\_list**[**j**].**due **>** key**.**due**)** **{**

CopyToDoItems**(&**tdList**->**item\_list**[**j**+**1**],&**tdList**->**item\_list**[**j**]);**

j **=** j **-** 1**;**

**}**

CopyToDoItems**(&**tdList**->**item\_list**[**j **+** 1**],&**key**);**

**}**

**}**

void setup\_wifi**()** **{**

delay**(**1**);**

// We start by connecting to a WiFi network

Serial**.**println**();**

Serial**.**print**(**"Connecting to "**);**

Serial**.**println**(**ssid**);**

**while** **(**status **!=** WL\_CONNECTED **&&** wifi\_wait**<**1**)** **{**

Serial**.**println**(**wifi\_wait**);**

status **=** WiFi**.**begin**(**ssid**,** password**);**

wifi\_wait**++;**

**if(**status **==** WL\_CONNECTED**)**

**{**

Serial**.**println**(**""**);**

Serial**.**println**(**"WiFi connected"**);**

Serial**.**println**(**"IP address: "**);**

Serial**.**println**(**WiFi**.**localIP**());**

unsigned char success **=** timeClient**.**forceUpdate**();**

current\_time **=** success**?** timeClient**.**getEpochTime**():**0**;**

wifi\_wait**=**0**;**

**return;**

**}**

**}**

**}**

JsonArray strToJsonArray**(**char**\*** jsonToDoList**){**

// parse a JSON array

deserializeJson**(**doc**,** jsonToDoList**);**

// extract the values

JsonArray array **=** doc**.**as**<**JsonArray**>();**

**return** array**;**

**}**

toDoItem**\*** getToDoItem**(**JsonArray array**,** unsigned char index**){**

JsonObject obj **=** array**[**index**];**

toDoItem**\*** tdItem **=** **(**toDoItem**\*)** calloc**(sizeof(**toDoItem**),** 1**);**

strcpy**(**tdItem**->**title**,**obj**[**"title"**]);**

tdItem**->**due **=** obj**[**"due"**];**

tdItem**->**yellow **=** obj**[**"yellow"**];**

tdItem**->**red **=** obj**[**"red"**];**

**return** tdItem**;**

**}**

String getFormattedTime**(**unsigned long t**,** unsigned long secs**=**0**)** **{**

unsigned long rawTime **=** secs **?** secs **:** t**;**

unsigned long hours **=** **(**rawTime **%** 86400L**)** **/** 3600**;**

String hoursStr **=** hours **<** 10 **?** "0" **+** String**(**hours**)** **:** String**(**hours**);**

unsigned long minutes **=** **(**rawTime **%** 3600**)** **/** 60**;**

String minuteStr **=** minutes **<** 10 **?** "0" **+** String**(**minutes**)** **:** String**(**minutes**);**

unsigned long seconds **=** rawTime **%** 60**;**

String secondStr **=** seconds **<** 10 **?** "0" **+** String**(**seconds**)** **:** String**(**seconds**);**

**return** hoursStr **+** ":" **+** minuteStr **+** ":" **+** secondStr**;**

**}**

String getFormattedDate**(**unsigned long t**,** unsigned long secs**)** **{**

unsigned long rawTime **=** **(**secs **?** secs **:** t**)** **/** 86400L**;** // in days

unsigned long days **=** 0**,** year **=** 1970**;**

uint8\_t month**;**

static const uint8\_t monthDays**[]={**31**,**28**,**31**,**30**,**31**,**30**,**31**,**31**,**30**,**31**,**30**,**31**};**

**while((**days **+=** **(**LEAP\_YEAR**(**year**)** **?** 366 **:** 365**))** **<=** rawTime**)**

year**++;**

rawTime **-=** days **-** **(**LEAP\_YEAR**(**year**)** **?** 366 **:** 365**);** // now it is days in this year, starting at 0

days**=**0**;**

**for** **(**month**=**0**;** month**<**12**;** month**++)** **{**

uint8\_t monthLength**;**

**if** **(**month**==**1**)** **{** // february

monthLength **=** LEAP\_YEAR**(**year**)** **?** 29 **:** 28**;**

**}** **else** **{**

monthLength **=** monthDays**[**month**];**

**}**

**if** **(**rawTime **<** monthLength**)** **break;**

rawTime **-=** monthLength**;**

**}**

String yearStr **=** String**(**year**);**

char year19**[**5**];**

strcpy**(**year19**,**yearStr**.**c\_str**());**

String monthStr **=** **++**month **<** 10 **?** "0" **+** String**(**month**)** **:** String**(**month**);** // jan is month 1

String dayStr **=** **++**rawTime **<** 10 **?** "0" **+** String**(**rawTime**)** **:** String**(**rawTime**);** // day of month

**return** monthStr **+** "/" **+** dayStr **+** "/" **+** **&**year19**[**3**]** **+** " " **+** getFormattedTime**(**secs **?** secs **:** 0**,** t**);**

**}**

void callback**(**char**\*** topic**,** byte**\*** payload**,** unsigned int length**)** **{**

Serial**.**print**(**"Message arrived ["**);**

Serial**.**print**(**topic**);**

Serial**.**print**(**"] "**);**

**for** **(**int i **=** 0**;** i **<** length**;** i**++)** **{**

Serial**.**print**((**char**)**payload**[**i**]);**

strcpy**(**jsonToDoList**,** **(**char**\*)**payload**);**

**}**

lcd**.**clear**();**

JsonArray array **=** strToJsonArray**(**jsonToDoList**);**

tdList**.**list\_length **=** 5**;**

**for(**int i**=**0**;** i**<**tdList**.**list\_length**;** i**++)**

**{**

toDoItem**\*** item **=** getToDoItem**(**array**,** i**);**

CopyToDoItems**(&**tdList**.**item\_list**[**i**],** item**);**

free**(**item**);**

**if(**tdList**.**item\_list**[**i**].**due **==** 0**)**

**{**

strcpy**(**tdList**.**item\_list**[**i**].**title**,**"NONE"**);**

tdList**.**item\_list**[**i**].**due **=** 2000000000**;**

tdList**.**item\_list**[**i**].**yellow **=** 0**;**

tdList**.**item\_list**[**i**].**red **=** 0**;**

**}**

**}**

passedDue **=** 0**;**

// printToDoList(&tdList);

insertionSort**(&**tdList**);**

// printToDoList(&tdList);

flash**.**write**(**tdList**);**

// timeClient.update();

unsigned char success **=** timeClient**.**forceUpdate**();**

current\_time **=** success**?** timeClient**.**getEpochTime**():**0**;**

Serial**.**println**();**

**}**

void reconnect**()** **{**

// Loop until we're reconnected

Serial**.**print**(**"Attempting MQTT connection..."**);**

// Attempt to connect

**if** **(**client**.**connect**(**USERNAME**))** **{**

Serial**.**println**(**"connected"**);**

// Once connected, subscribe to input channel

//client.subscribe(SUBSCRIBE\_TOPIC);

//client.publish(PUBLISH\_TOPIC,USERNAME);

**}** **else** **{**

Serial**.**print**(**"failed, rc="**);**

Serial**.**print**(**client**.**state**());**

Serial**.**println**(**" try again in 5 seconds"**);**

// Wait 5 seconds before retrying

// delay(5000);

**}**

**}**

void loop**()** **{**

**if(**millis**()** **-** lastCheckConnected **>=** 1000**){**

status **=** WiFi**.**status**();**

**if(**status **!=** WL\_CONNECTED **&&** wifi\_counter**>=**10**)**

**{**

wifi\_wait **=** 0**;**

wifi\_counter **=** 0**;**

setup\_wifi**();**

**}**

**else** **if(**status **==** WL\_CONNECTED**)**

**{**

**if(!**client**.**connected**())** **{**

reconnect**();**

client**.**subscribe**(**TODOLIST\_TOPIC**);**

**}**

client**.**loop**();**

**}**

lastCheckConnected **=** millis**();**

scroll\_counter**++;**

wifi\_counter**++;**

time\_update**++;**

**}**

**if(**time\_update**%**60 **==** 0**)**

**{**

time\_update **=** 0**;**

**if(**status **==** WL\_CONNECTED**)**

**{**

// timeClient.update();

unsigned char success **=** timeClient**.**forceUpdate**();**

current\_time **=** success**?** timeClient**.**getEpochTime**():**0**;**

**}**

**else**

current\_time **=** 0**;**

**}**

current\_time **=** **(**status **==** WL\_CONNECTED**)?** timeClient**.**getEpochTime**():**0**;**

**if(**digitalRead**(**button**)** **==** HIGH **&&** **!**passedDue**)**

**{**

scroll**++;**

scroll\_counter **=** 1**;**

**if(**scroll**>**4**)** scroll **=** 0**;**

lcd**.**clear**();**

delay**(**200**);**

**}**

//counter to return to the first element if not scrolled for 60 seconds

**if(**scroll\_counter**%**60**==**0**)**

**{**

scroll\_counter **=** 0**;**

scroll **=** 0**;**

**}**

**if(!**passedDue**)**

**{**

lcd**.**setCursor**(**0**,**0**);**

lcd**.**print**(**tdList**.**item\_list**[**scroll**].**title**);**

lcd**.**setCursor**(**0**,**1**);**

lcd**.**print**(**getFormattedDate**(**tdList**.**item\_list**[**scroll**].**due**));**

**}**

**else**

**{**

tone**(**6**,** 1047**,** piezo**);**

lcd**.**setCursor**(**0**,**0**);**

lcd**.**print**(**tdList**.**item\_list**[**0**].**title**);**

lcd**.**setCursor**(**0**,**1**);**

lcd**.**print**(**"Deadline passed!"**);**

**if(**digitalRead**(**button**)==**HIGH**)**

**{**

printToDoList**(&**tdList**);**

passedDue **=** 0**;**

strcpy**(**tdList**.**item\_list**[**0**].**title**,**"NONE"**);**

tdList**.**item\_list**[**0**].**due **=** 2000000000**;**

tdList**.**item\_list**[**0**].**yellow **=** 0**;**

tdList**.**item\_list**[**0**].**red **=** 0**;**

printToDoList**(&**tdList**);**

insertionSort**(&**tdList**);**

printToDoList**(&**tdList**);**

lcd**.**clear**();**

scroll **=** 0**;**

delay**(**200**);**

**}**

**}**

**if(**current\_time **<** tdList**.**item\_list**[**0**].**due **-** tdList**.**item\_list**[**0**].**yellow**)**

**{**

digitalWrite**(**yellow**,**LOW**);**

digitalWrite**(**green**,**HIGH**);**

digitalWrite**(**red**,**LOW**);**

**}**

**else** **if(**current\_time **>=** tdList**.**item\_list**[**0**].**due **-** tdList**.**item\_list**[**0**].**yellow**)**

**{**

**if(**current\_time **>=** tdList**.**item\_list**[**0**].**due **-** tdList**.**item\_list**[**0**].**red**)**

**{**

**if(**current\_time **>=** tdList**.**item\_list**[**0**].**due **&&** **!**passedDue**)**

**{**

lcd**.**clear**();**

passedDue **=** 1**;**

**}**

**else**

**{**

digitalWrite**(**yellow**,**LOW**);**

digitalWrite**(**green**,**LOW**);**

digitalWrite**(**red**,**HIGH**);**

**}**

**}**

**else**

**{**

digitalWrite**(**yellow**,**HIGH**);**

digitalWrite**(**green**,**LOW**);**

digitalWrite**(**red**,**LOW**);**

**}**

**}**

**}**