Database and Reader Configuration

The goal of this document is to detail the storage of data collected from the reader. For the purposes of this project, only the EPC codes were scanned and captured, but all data from tags such as the wisp could be collected and stored in a similar fashion. The flow chart below describes the process:

Reader gets tag info → ESP32 collects the data → HTTP Request to Web Server → PHP Script → MySQL

We're going to assume we already have the data that we need to input and begin the setup of the actual database.

Database Configuration

1. Download XAMPP. https://www.apachefriends.org/download.html
   XAMPP is an open-source web server stack that stands for Apache HTTP server, MariaDB (MySQL) database, PHP, and Perl. We are going to use XAMPP as a local server to host our information.
2. Once downloaded, open XAMPP and start both the Apache and MySQL Modules, the other three aren't used.
3. Open the Shell on the XAMPP control panel.
4. Login once using the root username and password by using the following in the shell:
   a. `mysql.exe -u root -p`
   b. You will then be prompted to enter a password, the password for the root account is: "your-root-password"
5. Now we are going to create a user account that we will use to access the database. The username is CarelessWISPer and the password is shep_123. These can be set to anything when first created. Type the following in the shell:
   a.
   ```
   CREATE USER 'CarelessWISPer'@'localhost' IDENTIFIED BY 'shep_123';
   GRANT ALL PRIVILEGES ON *.* TO 'CarelessWISPer'@'localhost' WITH GRANT OPTION;
   FLUSH PRIVILEGES;
   ```
6. To login to our newly created account, quit the MariaDB editor and reopen, then type:
   a. `Mysql.exe -u CarelessWISPer -p`
   b. Use 'shep_123' as the password.
7. Since we will be storing the epc values from the tags, we will create a database and table to store them. First, create a database called db_epc by entering the following into the shell:
   a.
   ```
   CREATE DATABASE db_epc CHARACTER SET = 'utf8'
   COLLATE='utf8_general_ci';
   ```
8. Now, create a table within the database that will actually store the information. First set which database we will be working in:
   a. `USE db_epc;`

9. Then, create the table named epc_vals (the name doesn't matter as long as everything is consistent across the ESP32 code, PHP script and MySQL Server).

Field	Type	Null	Key	Default	Extra
tag_id	int(10) unsigned	NO	PRI	NULL	auto_increment
product_name	varchar(40)	YES		NULL	
tag_epc	varchar(36)	YES		NULL	
last_edited	timestamp(6)	NO		current_timestamp(6)	on update current_timestamp(6)
tag_state	varchar(30)	YES		NULL	
entry_date	datetime(6)	YES		NULL	

```
CREATE TABLE epc_vals(
       tag_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
       product_name VARCHAR(40),
       tag_epc VARCHAR(36),
       last_edited TIMESTAMP(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6) ON UPDATE
       CURRENT_TIMESTAMP(6),
       tag_state VARCHAR(30),
       entry_date DATETIME(6),
       PRIMARY KEY (tag_id));
```

10. Now that we have a place to store our values, we can look into actually inserting them by writing a php script. Open a text editor and paste the following code, there should also be a php file included in the folder where this document was found that contains this code.

```
<?php

if(isset($_GET["epc"])) {
  $epc = $_GET["epc"];

  $servername = "localhost";
  $username = "CarelessWISPer";
  $password = "shep_123";
  $dbname = "db_epc";
  $timeout = "20";
  // Create connection
  $conn = new mysqli($servername, $username, $password, $dbname);
  // Check connection
  if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
  }
  $sql = "CALL insert_data($epc,$timeout)";
  if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
  } else {
    echo "Error: " . $sql . " => " . $conn->error;
  }

  $conn->close();
} else {
  echo "epc is not set";
}
?>
```

Save the code from above as a .php file and make sure that it is in the following directory:

a. C:\xampp\htdocs

11. One thing to note is that in the php file, there is a line that reads:
    a. `$sql = “CALL insert_data($epc,$timeout)”;`

    This is a call to a stored procedure in the database, you will need to make this function in order to make sure this works properly, to do so, paste the following code into the MySQL shell:

```
DELIMITER //
CREATE PROCEDURE insert_data(parameter_tag_epc VARCHAR(36), parameter_timeout INTEGER)
 MODIFIES SQL DATA
 BEGIN
 INSERT INTO epc_vals (tag_epc,entry_date,tag_state) VALUES (parameter_tag_epc,NOW(),'Recent Batch');
 DELETE FROM epc_vals WHERE entry_date < NOW() - INTERVAL parameter_timeout MINUTE;
 UPDATE epc_vals SET tag_state = 'Old Batch' WHERE entry_date < NOW() - INTERVAL 70 SECOND;
 UPDATE epc_vals SET product_name = 'CHEEZIT' WHERE LEFT(tag_epc,11) = '22600295125';
UPDATE epc_vals SET product_name = 'MT DEW' WHERE LEFT(tag_epc,11) = '16026516616';
 CREATE OR REPLACE TABLE product_info SELECT product_name, tag_epc,entry_date, COUNT(*) times_item_was_seen FROM
epc_vals GROUP BY tag_epc ORDER BY entry_date DESC;
 END;//
DELIMITER ;
```

A few things to note, specifically with the lines marked yellow. These are only used as a way to distinguish between our specific tags and organize the data we had, they are not necessary to the functionality of the database.

12. Then, create an Arduino Sketch and place the following code into three separate files. Make sure to change the wifi and server settings to your appropriate settings.

Main Code:

```
/*
  Epaper Display Scan for Gen2 EPC Values and Update Database
  Authors: DJ Hansen, Joe Harrison, Jon Mark Long, Daniel Higgins
  Date: November 29th, 2021
  About: Electronic Wireless Shopping Label System code including Gen 2 Reader, MySQL Database Connection, and
  Wireless Updating of EPaper Display

*/


/*

**************************************************************************************************************
*************
 *                                        Preproccesor Directives, MACROS, and header files
*

**************************************************************************************************************
*************
 */

// processor predirectives and MACROS
#define use_soft_serial 0                // predirective to control whether using software or hardware serial port,
if set to 1, then switch on M6E needs to be set to SW
#define DEBUG 0                          // 0 : no messages  1 : request sending and receiving
#define EPC_COUNT 20                     // how many tags are expected in the area, can reasonably go up to around
100
#define NANOREGION REGION_NORTHAMERICA    //which region the reader operates in, check
https://github.com/sparkfun/Simultaneous_RFID_Tag_Reader for other regions
#define ALWAYS 1

// Reader header files
#include "SparkFun_UHF_RFID_Reader.h"     //Library for controlling the M6E Nano module

#if use_soft_serial
  #include <SoftwareSerial.h>             //Used for transmitting to the device
  SoftwareSerial softSerial(2, 3);        //RX, TX
#else
  #define NanoSerial Serial               // define the serial port to use (E.g. softSerial, Serial1 etc)
#endif

// Database header files
#include <WiFi.h>
#include <HTTPClient.h>

// ePaper header files
#include <WiFiClient.h>
#include <WebServer.h>
#include <ESPmDNS.h>
#include <Update.h>
#include <SPI.h>


/**************************************************************************************************************
*********
 *                     Global Variables and Objects needed for Reader and Database Functionality
*

**************************************************************************************************************
********
 */


uint8_t EPC_recv[EPC_COUNT][12];                //stores unique tag values in a session
const unsigned long time_pd_wait = 60000;       // how long we should wait in between scan sessions (ms)
const unsigned long time_pd_scan = 30000;       // scan duration (ms)
unsigned long prev_time = 0;                    // timer wait update
```

```
unsigned long prev_scan_time = 0;               // timer scan update
bool onetime = true;                            // one time flag
bool firstTime = true;                          // one time flag
unsigned long scan_time = 0;                    // used to store millis()



const char WIFI_SSID[] = "ncsu";               // wifi network name
const char WIFI_PASSWORD[] = "";               // wifi password
String HOST_NAME = "http://10.155.25.216:80";  // PC's IP address, use port 80
String PATH_NAME   = "/insert_epc.php";        // php file placed in C:\xampp\htdocs
String queryString;                            // query to send to database
String final_epc[EPC_COUNT];                   // string epc values

const unsigned long update_db = 30000;          //database update time (ms)
unsigned long prev_time_db = 0;                // timer update
uint8_t first_flag = 0;                        // one time flag

int httpCode;                                  // stores code from http connection attempt
String payload;                                //

// Object Instances
RFID nano;                                     //Create instance for reader
HTTPClient http;                              //Create instance for http connection



/*
 *
 *
**********************************************************************************************************************
**
 * Everything from this point until line 365 is used for Updating and Flashing an image onto the Epaper Display
*
 *
**********************************************************************************************************************
**
 */

bool add_image_done = false;
const char* host = "esp32";
const char* ssid = "ncsu";
const char* password = "";

#define EPD_154 0 //1.54 inch PDi EPD (iTC)
#define EPD_213 1 //2.13 inch PDi EPD (iTC)
#define EPD_266 2 //2.66 inch PDi EPD (iTC)
#define EPD_271 3 //2.71 inch PDi EPD (iTC)
#define EPD_287 4 //2.87 inch PDi EPD (iTC)
#define EPD_370 5 //3.70 inch PDi EPD (iTC)
#define EPD_420 6 //4.20 inch PDi EPD (iTC)
#define EPD_437 7 //4.37 inch PDi EPD (iTC)

long image_data_size[] = { 2888, 2756, 5624, 5808, 4736, 12480, 15000, 10560 };  //followed by the index above

#define PDI_EPD_Size EPD_266

#if (PDI_EPD_Size >= 5)                                                          //3.70", 4.20", 4.37"
    uint8_t register_data[] = { 0x00, 0x0e, 0x19, 0x02, 0x0f, 0x89 };           //0x00, soft-reset, temperature,
active temp., PSR0, PSR1
#else                                                                            //other small sizes
    uint8_t register_data[] = { 0x00, 0x0e, 0x19, 0x02, 0xcf, 0x8d };
#endif

#if(PDI_EPD_Size==2)//2.66"
    #include "image_266_296x152_BW.c"
    #define BW_monoBuffer        (uint8_t *) & image_266_296x152_BW_monoTest
    #define BW_monoBuffer1        (uint8_t *) & image_266_296x152_BW_mono1
    #define BW_monoBuffer2        (uint8_t *) & image_266_296x152_BW_mono2
    #define BW_monoBuffer3        (uint8_t *) & image_266_296x152_BW_mono3
    #define BW_monoBuffer4        (uint8_t *) & image_266_296x152_BW_mono4
    #define BW_0x00Buffer        (uint8_t *) & image_266_296x152_BW_0x00
#elif(PDI_EPD_Size==3) //2.71"
```

```
    #include "image_data\2.71\image_271_264x176_BW.c"
    #include "image_data\2.71\image_271_264x176_BWR.c"
    #define BW_monoBuffer        (uint8_t *) & image_271_264x176_BW_mono
    #define BW_0x00Buffer        (uint8_t *) & image_271_264x176_BW_0x00
    #define BWR_blackBuffer      (uint8_t *) & image_271_264x176_BWR_blackBuffer
    #define BWR_redBuffer        (uint8_t *) & image_271_264x176_BWR_redBuffer

#endif

#if defined(ENERGIA)                                                    // Valid pins for LaunchPad on
Energia
#define SCL_PIN 7                                                       // EXT3 board J4 pin 2 SCK
#define BUSY_PIN 11                                                     // EXT3 board J4 pin 3 BUSY
#define DC_PIN 12                                                       // EXT3 board J4 pin 4 D/C
#define RESET_PIN 13                                                    // EXT3 board J4 pin 5 RST (RESET)
#define SDA_PIN 15                                                     // EXT3 board J4 pin 7 MOSI
#define CS_PIN 19                                                       // EXT3 board J4 pin 9 ECSM (EPD CS
Master)

// SPI protocl setup
void sendIndexData( uint8_t index, const uint8_t *data, uint32_t len ) {
  SPI.begin ();
  SPI.setDataMode(SPI_MODE3);
  SPI.setClockDivider(SPI_CLOCK_DIV32);
  SPI.setBitOrder(MSBFIRST);
  digitalWrite( DC_PIN, LOW );                                          //DC Low
  digitalWrite( CS_PIN, LOW );                                          //CS Low
  delayMicroseconds(500);
  SPI.transfer( index );
  delayMicroseconds(500);
  digitalWrite( CS_PIN, HIGH );                                          //CS High
  digitalWrite( DC_PIN, HIGH );                                          //DC High
  digitalWrite( CS_PIN, LOW );                                          //CS Low
  delayMicroseconds(500);
  for ( int i = 0; i < len; i++ ) SPI.transfer( data[ i ] );
  delayMicroseconds(500);
  digitalWrite( CS_PIN, HIGH );                                         //CS High
}

#else // Valid pins for Arduino board, like M0 Pro
#define SCL_PIN 18                                                      // EXT3 board J4 pin 2 SCK
#define BUSY_PIN 15                                                     // EXT3 board J4 pin 3 BUSY
#define DC_PIN 2                                                        // EXT3 board J4 pin 4 D/C
#define RESET_PIN 4                                                     // EXT3 board J4 pin 5 RST (RESET)
#define SDA_PIN 23                                                      // EXT3 board J4 pin 7 MOSI
#define CS_PIN 21                                                      // EXT3 board J4 pin 9 ECSM (EPD CS Master)


// Software SPI setup
void softwareSpi( uint8_t data ) {
  for ( int i = 0; i < 8; i++ ) {
    if ((( data >> (7 - i) ) & 0x01 ) == 1 ) digitalWrite( SDA_PIN, HIGH );
    else digitalWrite( SDA_PIN, LOW );
    digitalWrite( SCL_PIN, HIGH );
    digitalWrite( SCL_PIN, LOW );
  }
}

// Software SPI protocl setup
void sendIndexData( uint8_t index, const uint8_t *data, uint32_t len ) {
  digitalWrite( DC_PIN, LOW );                                          //DC Low
  digitalWrite( CS_PIN, LOW );                                          //CS Low
  softwareSpi( index );
  digitalWrite( CS_PIN, HIGH );                                         //CS High
  digitalWrite( DC_PIN, HIGH );                                         //DC High
  digitalWrite( CS_PIN, LOW );                                          //CS High
  for ( int i = 0; i < len; i++ ) softwareSpi( data[ i ] );
  digitalWrite( CS_PIN, HIGH );                                         //CS High
}
#endif

//variabls for blinking an LED with Millis
```

```
const int led = 2;                                                      // ESP32 Pin to which onboard LED is
connected
unsigned long previousMillis = 0;                                       // will store last time LED was updated
const long interval = 1000;                                             // interval at which to blink
(milliseconds)
int ledState = LOW;                                                     // ledState used to set the LED

WebServer server(80);

/*
 * Login page
 */

const char* loginIndex =
 "<form name='loginForm'>"
    "<table width='20%' bgcolor='A09F9F' align='center'>"
        "<tr>"
            "<td colspan=2>"
                "<center><font size=4><b>ESP32 Login Page</b></font></center>"
                "<br>"
            "</td>"
            "<br>"
            "<br>"
        "</tr>"
        "<tr>"
             "<td>Username:</td>"
             "<td><input type='text' size=25 name='userid'><br></td>"
        "</tr>"
        "<br>"
        "<br>"
        "<tr>"
            "<td>Password:</td>"
            "<td><input type='Password' size=25 name='pwd'><br></td>"
            "<br>"
            "<br>"
        "</tr>"
        "<tr>"
            "<td><input type='submit' onclick='check(this.form)' value='Login'></td>"
        "</tr>"
    "</table>"
"</form>"
"<script>"
    "function check(form)"
    "{"
    "if(form.userid.value=='admin' && form.pwd.value=='admin')"
    "{"
    "window.open('/serverIndex')"
    "}"
    "else"
    "{"
    " alert('Error Password or Username')/*displays error message*/"
    "}"
    "}"
"</script>";

/*
 * Server Index Page
 */

const char* serverIndex =
"<script src='https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js'></script>"
"<form method='POST' action='#' enctype='multipart/form-data' id='upload_form'>"
   "<input type='file' name='update'>"
        "<input type='submit' value='Update'>"
    "</form>"
 "<div id='prg'>progress: 0%</div>"
 "<script>"
  "$('form').submit(function(e){"
  "e.preventDefault();"
  "var form = $('#upload_form')[0];"
  "var data = new FormData(form);"
  " $.ajax({"
  "url: '/update',"
```

```
  "type: 'POST',"
  "data: data,"
  "contentType: false,"
  "processData:false,"
  "xhr: function() {"
  "var xhr = new window.XMLHttpRequest();"
  "xhr.upload.addEventListener('progress', function(evt) {"
  "if (evt.lengthComputable) {"
  "var per = evt.loaded / evt.total;"
  "$('#prg').html('progress: ' + Math.round(per*100) + '%');"
  "}"
  "}, false);"
  "return xhr;"
  "},"
  "success:function(d, s) {"
  "console.log('success!')"
 "},"
 "error: function (a, b, c) {"
 "}"
 "});"
 "});"
 "</script>";

void add_image(void){
  pinMode( SCL_PIN, OUTPUT );
  pinMode( SDA_PIN, OUTPUT );
  pinMode( CS_PIN, OUTPUT );
  pinMode( DC_PIN, OUTPUT );
  pinMode( RESET_PIN, OUTPUT );
  pinMode( BUSY_PIN, INPUT );                                          //All Pins 0
  delay( 5 );
  digitalWrite( RESET_PIN, HIGH );                                      //RES# = 1
  delay( 5 );
  digitalWrite( RESET_PIN, LOW );
  delay( 10 );
  digitalWrite( RESET_PIN, HIGH );
  delay( 5 );
  digitalWrite( CS_PIN, HIGH );                                         //CS# = 1

  sendIndexData( 0x00, &register_data[1], 1 );                          //Soft-reset
  while( digitalRead( BUSY_PIN ) != HIGH );

  sendIndexData( 0xe5, &register_data[2], 1 );                          //Input Temperature: 25C
  sendIndexData( 0xe0, &register_data[3], 1 );                          //Active Temperature
  sendIndexData( 0x00, &register_data[4], 2 );                           //PSR

  // Send 1st image data for black and white colors
  sendIndexData( 0x10, BW_monoBuffer1, image_data_size[PDI_EPD_Size] ); //First frame: black frame where 1=black and
0=white pixel
  sendIndexData( 0x13, BW_0x00Buffer, image_data_size[PDI_EPD_Size] );  //Second frame: all 0x00

  sendIndexData( 0x04, &register_data[0], 1 );                          //Power on
  while( digitalRead( BUSY_PIN ) != HIGH );
  sendIndexData( 0x12, &register_data[0], 1 );                          //Display Refresh
  while( digitalRead( BUSY_PIN ) != HIGH );

  delay( 5000 );

// Send 2nd image data for black and white colors
  sendIndexData( 0x10, BW_monoBuffer2, image_data_size[PDI_EPD_Size] );  //First frame: black frame where 1=black
and 0=white pixel
  sendIndexData( 0x13, BW_0x00Buffer, image_data_size[PDI_EPD_Size] );  //Second frame: all 0x00

  sendIndexData( 0x04, &register_data[0], 1 );                          //Power on
  while( digitalRead( BUSY_PIN ) != HIGH );
  sendIndexData( 0x12, &register_data[0], 1 );                         //Display Refresh
  while( digitalRead( BUSY_PIN ) != HIGH );

  delay( 5000 );

  // Send 3rd image data for black and white colors
  sendIndexData( 0x10, BW_monoBuffer3, image_data_size[PDI_EPD_Size] );  //First frame: black frame where 1=black
and 0=white pixel
```

```
  sendIndexData( 0x13, BW_0x00Buffer, image_data_size[PDI_EPD_Size] );  //Second frame: all 0x00

  sendIndexData( 0x04, &register_data[0], 1 );                          //Power on
  while( digitalRead( BUSY_PIN ) != HIGH );
  sendIndexData( 0x12, &register_data[0], 1 );                          //Display Refresh
  while( digitalRead( BUSY_PIN ) != HIGH );

  delay( 5000 );

  //Turn-off DC/DC
  sendIndexData( 0x02, &register_data[0], 1 );                          //Turn off DC/DC
  while( digitalRead( BUSY_PIN ) != HIGH );
  digitalWrite( DC_PIN, LOW );
  digitalWrite( CS_PIN, LOW );
  digitalWrite( SDA_PIN, LOW );
  digitalWrite( SCL_PIN, LOW );
  digitalWrite( BUSY_PIN, LOW );
  delay( 150 );
  digitalWrite( RESET_PIN, LOW );
}

 // End Joe stuff


/*
 *
**********************************************************************************************************************
**********
 *  Function: Setup
 *  Author: DJ Hansen, Joe Harrison, Jon Mark Long, Daniel Higgins
 *  Date: 11/29/2021
 *  About:
 *  Runs once before going into the Always loop.
 *  Sets up WiFi, Server, and Serial Connections
 *
**********************************************************************************************************************
**********
 */

void setup()
{

  Serial.begin(115200);

  // Connect to WiFi network

  WiFi.begin(ssid, password);
  Serial.println("");

  // Wait for connection

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

/**********************************************************************************************************************
**********/
  // Begin Server Connection Setup for EPaper Display

  /*use mdns for host name resolution*/
  if (!MDNS.begin(host)) { //http://esp32.local
    Serial.println("Error setting up MDNS responder!");
    while (1) {
      delay(1000);
    }
  }
  Serial.println("mDNS responder started");
```

```
  /*return index page which is stored in serverIndex */
  server.on("/", HTTP_GET, []() {
    server.sendHeader("Connection", "close");
    server.send(200, "text/html", loginIndex);
  });
  server.on("/serverIndex", HTTP_GET, []() {
    server.sendHeader("Connection", "close");
    server.send(200, "text/html", serverIndex);
  });
  /*handling uploading firmware file */
  server.on("/update", HTTP_POST, []() {
    server.sendHeader("Connection", "close");
    server.send(200, "text/plain", (Update.hasError()) ? "FAIL" : "OK");
    ESP.restart();
  }, []() {
    HTTPUpload& upload = server.upload();
    if (upload.status == UPLOAD_FILE_START) {
      Serial.printf("Update: %s\n", upload.filename.c_str());
      if (!Update.begin(UPDATE_SIZE_UNKNOWN)) {                                 //start with max available size
        Update.printError(Serial);
      }
    } else if (upload.status == UPLOAD_FILE_WRITE) {
      /* flashing firmware to ESP*/
      if (Update.write(upload.buf, upload.currentSize) != upload.currentSize) {
        Update.printError(Serial);
      }
    } else if (upload.status == UPLOAD_FILE_END) {
      if (Update.end(true)) {                                                   //true to set the size to the current
progress
        Serial.printf("Update Success: %u\nRebooting...\n", upload.totalSize);
      } else {
        Update.printError(Serial);
      }
    }
  });
  server.begin();
  // End Server Connection Setup for EPaper Display

/*************************************************************************************************************
***********/

  // Begin Reader Setup
  init_array();                                                                 // initializes storage values of EPC
to all 0
  Serial.begin(115200);                                                         // defined elsewhere, can probably
remove
  while (!Serial);                                                              //Wait for the serial port to come
online

  Serial.println(F("Ready to Begin Program"));                                  //If this prints, then no wiring
issues found
  if (DEBUG) nano.enableDebugging(Serial);

                                                                                // SetupNano establishes connection
between ESP32 and M6E Nano Reader

  if (setupNano(38400) == false)                                                //Configure nano to run at 38400bps
  {
    Serial.println(F("Module failed to respond. Please check wiring."));
    while (1); //Freeze!
  }

  nano.setRegion(NANOREGION);                                                  //Set to the right region

  nano.setReadPower(500);                                                     //5.00 dBm. Higher values may caues USB
port to brown out
                                                                              //Max Read TX Power is 27.00 dBm and may
cause temperature-limit throttling
  // End Reader Setup


  // Database setup
  connect_to_wifi();
```

```
/*****************************************************************************************************************
**************/
}


/*
 *
*****************************************************************************************************************
************************
 *  Function: loop
 *  Author: DJ Hansen and Joe Harrison
 *  Date: 11/29/2021
 *  From: North Carolina State University
 *  About:
 *  This loop runs continuously, checking first to make sure ePaper display is properly setup,
 *  Then goes into loop of scanning for time_pd_scan and then waiting for time_pd_wait
 *  Any Use of Serial.println is used for debugging purposes only and does not affect the outcome of the program
 *
 *
 *
*****************************************************************************************************************
************************
 */
void loop()
{
  unsigned long curr_time = millis();                          // gets current run time
/*****************************************************************************************************************
****************************/
// Begin Epaper Section of Loop

  server.handleClient();
  delay(5);
  if(!add_image_done){
    add_image();
    add_image_done = true;
  }

// End Epaper Section of loop
/*****************************************************************************************************************
****************************/
// Begin Reader and Database Loop

  if(((curr_time - prev_time) >= time_pd_wait) || onetime){      // This checks for two things: either we are going
into this loop for the first time,
    onetime = false;                                            // or we have waited the designated amount of time.
    clear_arrays();
    nano.startReading();                                        //Begin scanning for tags
    Serial.println(F("Done waiting"));
    prev_scan_time = millis();
    scan_time = millis();
    scan_time += 1;

    while(scan_time - prev_scan_time < time_pd_scan){           // Stay in this loop until we have been in it for
time_pd_scan ms
      scan_time = millis();

      if (nano.check() == true)                                 //Check to see if any new data has come in from
module
      {
        byte responseType = nano.parseResponse();               //Break response into tag ID, RSSI, frequency, and
timestamp

        if (responseType == RESPONSE_IS_KEEPALIVE)              // Haven't found anything yet......
        {
          Serial.println(F("Scanning"));
        }
        else if (responseType == RESPONSE_IS_TAGFOUND)          // Great! We found a Tag!
        {
                                                                //If we have a full record we can pull out the fun
bits
          int rssi = nano.getTagRSSI();                         //Get the RSSI for this tag read
          long freq = nano.getTagFreq();                        //Get the frequency this tag was detected at
```

```
          long timeStamp = nano.getTagTimestamp();              //Get the time this was read, (ms) since last keep-
alive message
          byte tagEPCBytes = nano.getTagEPCBytes();             //Get the number of bytes of EPC from response

      if(check_array(nano.msg)){                                // Check to see if we have already found this tag in
this session, don't do anything if we have
          }else{
          add_array(nano.msg);                                 // If we haven't then store it
                                                               //Print EPC bytes, this is a subsection of bytes from
the response/msg array

          Serial.print(F(" epc["));                            // This code is just used to print it to serial
output
          for (byte x = 0 ; x < tagEPCBytes ; x++)
          {
            if (nano.msg[31 + x] < 0x10) Serial.print(F("0"));
            Serial.print(nano.msg[31 + x], HEX);
            Serial.print(F(" "));
          }
          Serial.print(F("]"));

          Serial.println();
          }
        }
        else if (responseType == ERROR_CORRUPT_RESPONSE)      // Bad CRC
        {
          Serial.println("Bad CRC");
        }
        else
        {
                                                               //Unknown response
          Serial.print("Unknown error");
        }
      }
    }
    firstTime = false;                                        // Shut Everything down and begin storing found tags in
database
    nano.stopReading();
    Serial.println(F("Done scanning"));
    delay(500);
    database();
    prev_time = curr_time;
  }
// End Reader and Database Loop
/**********************************************************************************************************************
************************/
}

/**********************************************************************************************************************
****************************
 * Function: setupNano
 * Author:   Nathan Seidle @ SparkFun Electronics
 * Date: October 3rd, 2016
 * From:  https://github.com/sparkfun/Simultaneous_RFID_Tag_Reader
 * About: Gracefully handles a reader that is already configured and already reading continuously
 * Because Stream does not have a .begin() we have to do this outside the library
 *

**********************************************************************************************************************
************************/

boolean setupNano(long baudRate)
{
  nano.begin(NanoSerial);                                  //Tell the library to communicate over serial port

                                                          //Test to see if we are already connected to a module
                                                          //This would be the case if the Arduino has been reprogrammed
and the module has stayed powered
  NanoSerial.begin(baudRate);                            //For this test, assume module is already at our desired baud
rate
  while(!NanoSerial);                                     //Wait for port to open
```

```
                                                //About 200ms from power on the module will send its firmware
version at 115200. We need to ignore this.
  while(NanoSerial.available()) NanoSerial.read();

  nano.getVersion();

  if (nano.msg[0] == ERROR_WRONG_OPCODE_RESPONSE)
  {                                             //This happens if the baud rate is correct but the module is
doing a ccontinuous read
    nano.stopReading();
    Serial.println(F("Module cont. reading. Asking it to stop..."));
    delay(1500);
  }

  else if (nano.msg[0] != ALL_GOOD)
  {                                             //The module did not respond so assume it's just been powered on
and communicating at 115200bps
    NanoSerial.begin(115200);                   //Start software serial at 115200
    nano.setBaud(baudRate);                     //Tell the module to go to the chosen baud rate. Ignore the
response msg

    NanoSerial.begin(baudRate);                 //Start the serial port, this time at user's chosen baud rate

    nano.getVersion();                          //Test the connection
    if (nano.msg[0] != ALL_GOOD) return (false);    //Something is not right
  }

  //The M6E has these settings no matter what
  nano.setTagProtocol();                        //Set protocol to GEN2

  nano.setAntennaPort();                       //Set TX/RX antenna ports to 1

  return (true);                               //We are ready to rock
}

/****************************************************************************************************************
*****************************
 * Function: init_array
 * Author:   DJ Hansen @ North Carolina State University
 * Date: 11/29/21
 * From:
 * About: Initializes all storage arrays to zero
 *

****************************************************************************************************************
************************/
void init_array(){
  int i;

  for(i=0;i<EPC_COUNT;i++){
    EPC_recv[i][0] = 0;
  }
}
/****************************************************************************************************************
*****************************
 * Function: check_array
 * Author:   DJ Hansen @ North Carolina State University
 * Date: 11/29/21
 * From:
 * About: Checks to see if the scanned tag has already been scanned in this section
 *

****************************************************************************************************************
************************/
bool check_array(uint8_t * msg){
  int i,j;
  int found;
  i=0;

  // as long as not end of list
  while(i < EPC_COUNT && EPC_recv[i][0] != 0) {
    found = 1;
    for (j = 0 ; j < 12 ; j++) {
```

```
      if (EPC_recv[ i ] [ j ] != msg[31 + j]) {
        found = 0;
        j = 12;
        i++;
      }
    }

    // if found
    if (found == 1) return 1;
  }
  return 0;
}
/*****************************************************************************************************************
****************************
 * Function: add_array
 * Author:   DJ Hansen @ North Carolina State University
 * Date: 11/29/21
 * From:
 * About: If this tag has not been scanned this section, then store it
 *

*****************************************************************************************************************
************************/
void add_array(uint8_t *msg) {
  int i,j;
  int found;
  i=0;

  // as long as not end of list
  while(i < EPC_COUNT && EPC_recv[i][0] != 0) {
    found = 1;
    for (j = 0 ; j < 12 ; j++) {
      if (EPC_recv[ i ] [ j ] != msg[31 + j]){
        found = 0;
        j = 12;
        i++;
      }
    }
    // if found
    if (found == 1) return;
  }

  if (i == EPC_COUNT) {
    Serial.print(F("Can not add more to array"));
    Serial.println();
    return;
  }
  // add to array
  for (j = 0 ; j < 12 ; j++) {
    EPC_recv[ i ] [ j ] = msg[31 + j];
  }
  Serial.print(F("Entry added"));
  Serial.println();
}
/*****************************************************************************************************************
****************************
 * Function: count_entries
 * Author:   DJ Hansen @ North Carolina State University
 * Date: 11/29/21
 * From:
 * About: Counts the number of tags scanned in a session
 *

*****************************************************************************************************************
************************/
int count_entries() {
  int i=0;

  while(i < EPC_COUNT) {
    if( EPC_recv[ i ] [ 0 ] == 0)  break;
    i++;
  }
  return i;
```

```
    }
    /*****************************************************************************************************************
    ****************************
     * Function: clear_arrays
     * Author:   DJ Hansen @ North Carolina State University
     * Date: 11/29/21
     * From:
     * About: same as init_array only this also sets the tag data sent to the database to a null string
     *

    *****************************************************************************************************************
    *************************/
    void clear_arrays(void){
      for(int i = 0; i < EPC_COUNT; i++){
        for(int j = 0; j < 12; j++){
         EPC_recv[i][j] = 0;
         final_epc[i] = "";
        }
      }
    }
```

DATABASE CODE

```
/*****************************************************************************************************************************************************
****
 * Function: database
 * Author:   DJ Hansen @ North Carolina State University
 * Date: 11/29/21
 * From:
 * About: connects to wifi, converts collected tag data to string, and sends an HTTP request to the apache web server setup using xampp.
 *

*****************************************************************************************************************************************************
*/
void database(){
  unsigned long curr_time_db = millis();
                                                                                  // setup wifi connection
  if(WiFi.status() != WL_CONNECTED){
    connect_to_wifi();
  }
    convert_epc_string();

    for(int h = 0; h < count_entries(); h++){
      queryString = "?epc=" + final_epc[h];
      http.begin(HOST_NAME + PATH_NAME + queryString); //HTTP
      httpCode = http.GET();


      if(httpCode > 0) {                                                           // httpCode will be negative on error
        if(httpCode == HTTP_CODE_OK) {
        payload = http.getString();
        Serial.println(payload);
        Serial.println(httpCode);
        Serial.println("here");
      } else {                                                                    // HTTP header has been send and Server response header has been
handled
          Serial.printf("[HTTP] GET... code: %d\n", httpCode);
      }
    } else {
        Serial.printf("[HTTP] GET... failed, error: %s\n", http.errorToString(httpCode).c_str());
    }

      http.end();

    }

    clear_arrays_db();
    prev_time_db = curr_time_db;
    if(!first_flag){
      first_flag = 1;
  }
}


/*****************************************************************************************************************************************************
****
 * Function: connect_to_wifi
 * Author:   DJ Hansen @ North Carolina State University
```

```
 * Date: 11/29/21
 * From:
 * About: sets up wifi connection
 *

**********************************************************************************************************************************************
*/
void connect_to_wifi(void){
      WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
    }
    Serial.println("");
    Serial.print("Connected to WiFi network with IP Address: ");
    Serial.println(WiFi.localIP());
}
/**********************************************************************************************************************************************
****
 * Function: convert_epc_string
 * Author:   DJ Hansen @ North Carolina State University
 * Date: 11/29/21
 * From:
 * About: simple integer array to string
 *

**********************************************************************************************************************************************
*/
void convert_epc_string(void){
  for(int i = 0; i < count_entries(); i++){
    for(int j = 0; j < 12; j++){
      final_epc[i] += String(EPC_recv[i][j]);
    }
  }
}
/**********************************************************************************************************************************************
****
 * Function: clear_arrays_db
 * Author:   DJ Hansen @ North Carolina State University
 * Date: 11/29/21
 * From:
 * About: same as clear_arrays

**********************************************************************************************************************************************
*/
void clear_arrays_db(void){
    for(int i = 0; i < EPC_COUNT; i++){
      for(int j = 0; j < 12; j++){
       EPC_recv[i][j] = 0;
       final_epc[i] = "";
      }
    }
}
```

IMAGE CODE

Paste the appropriate image_266_296x152_BW.c file converted into hexadecimal format in this file.