بسم الله الرحمن الرحيم

ELECTROLYZER

PROCESS CONTROL SYSTEM

Produced by: Nour Karim

Raja Murad

Last update: Monday, September 20, 2021

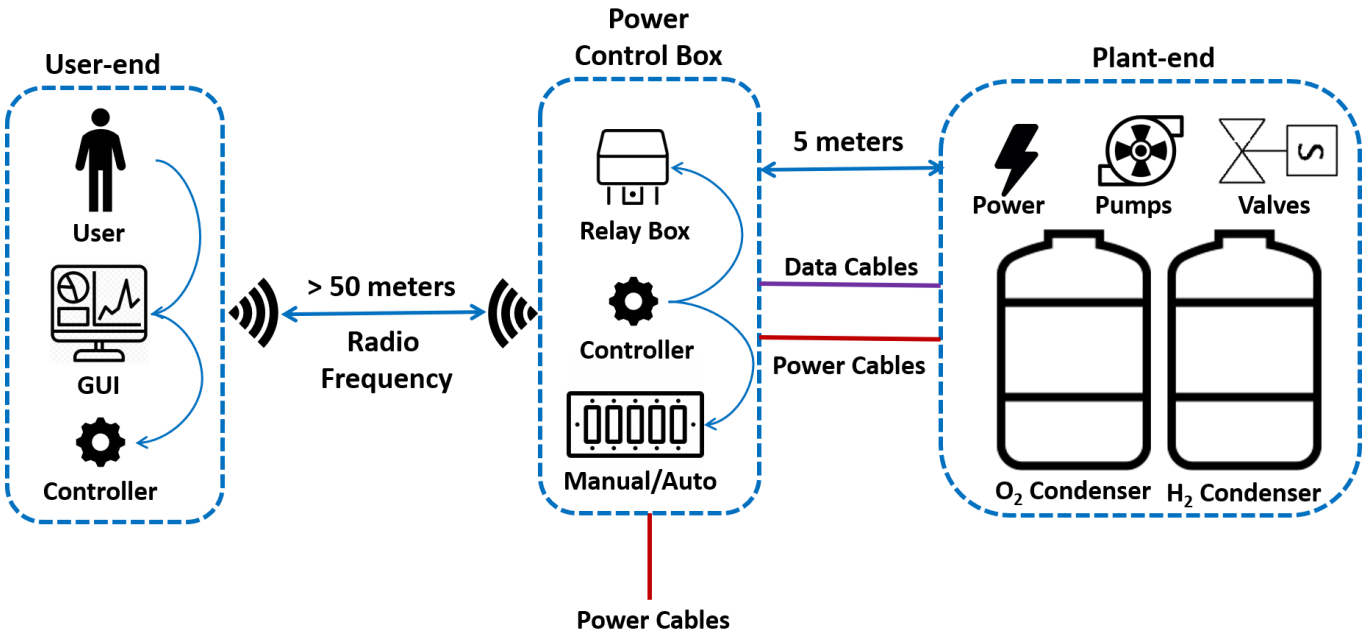
Table of Contents

1. Requirements and Overview

The electrolyzer process control system to be implemented should have the following technical requirements:

- The user must be able to switch between manual and auto modes.
- The user must be able to control the electrolyzer plant from a minimum distance of 50 meters.
- The user must be able to monitor different sensor data on a well-organized GUI.
- The user must be able to select the valves' opening angles from the GUI.
- The system must have a 2-way communication channel.

In other words, the system functionality can be summarized by the following graphical representation:



The system can be divided into 3 subcategories; The plant side where the electrolyzer setup is set, the power control unit that handles all the control procedure, and the user side where all the controls are set.

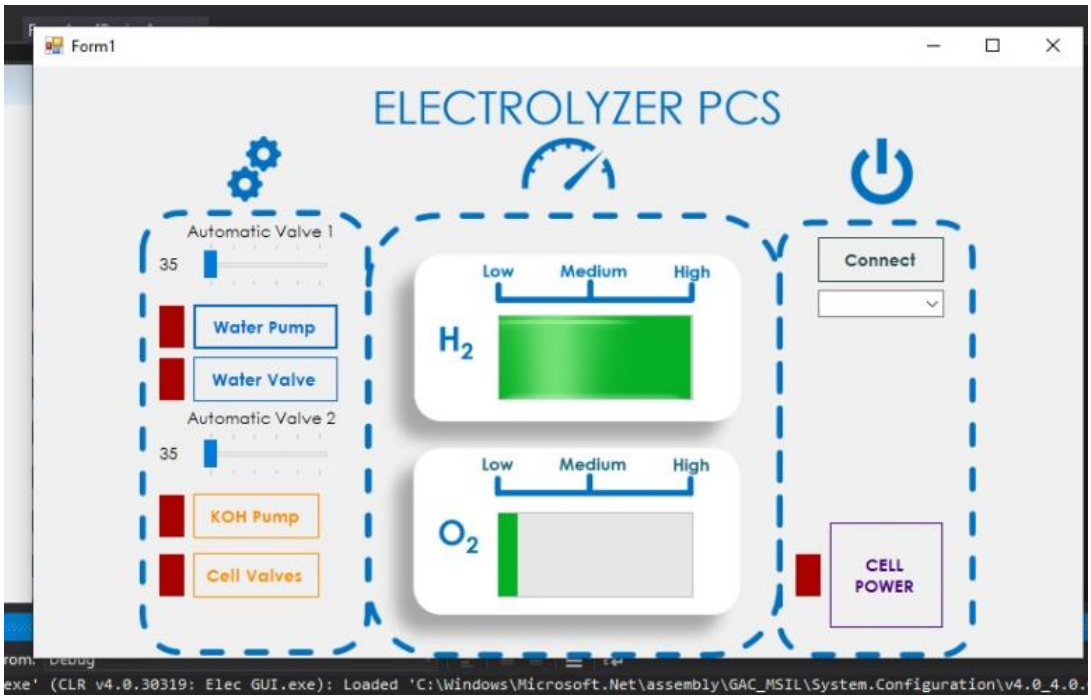
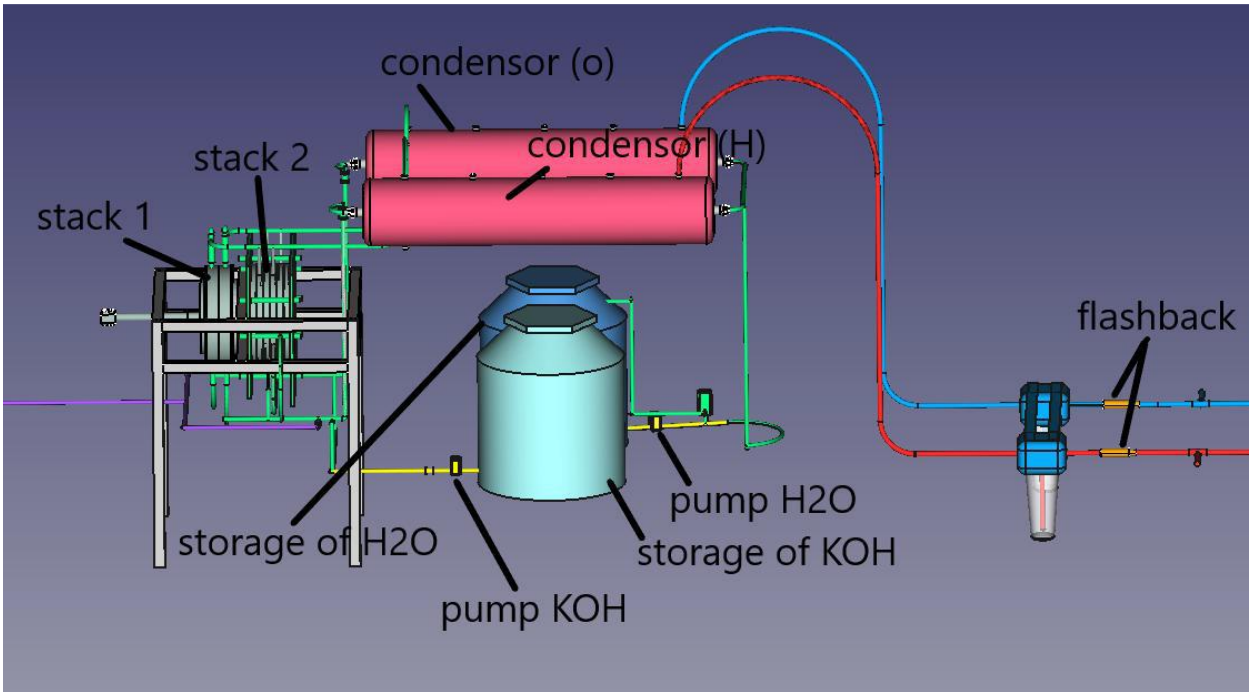


Figure 1. User-end

Figure 2. power control box

Figure 3. plant-end

2. Plant-End


condensor (o)
condensor (H)
stack 2
stack 1
flashback
pump H2O
storage of KOH
storage of H2O
pump KOH

2.1. Automatic valves

An Automatic valve is an electromechanical device that uses an electric current to generate a magnetic field and thus actuate a solenoid that controls the opening of the fluid flow in a valve.

2.1.1. Water Automatic valve

The following table presents the characteristics needed of water Automatic valve:

Name	Type	Port Number	Control type
Electrolysis cell Water Discharge	Direct-operated Automatic valve	3 (3-way)	Supply Voltage
Gas condenser Water Discharge	Servo motor	2 (2-way)	Supply Voltage + PWM

Figure 4. 3-way Automatic valve

Figure 5. 2-way Automatic valve

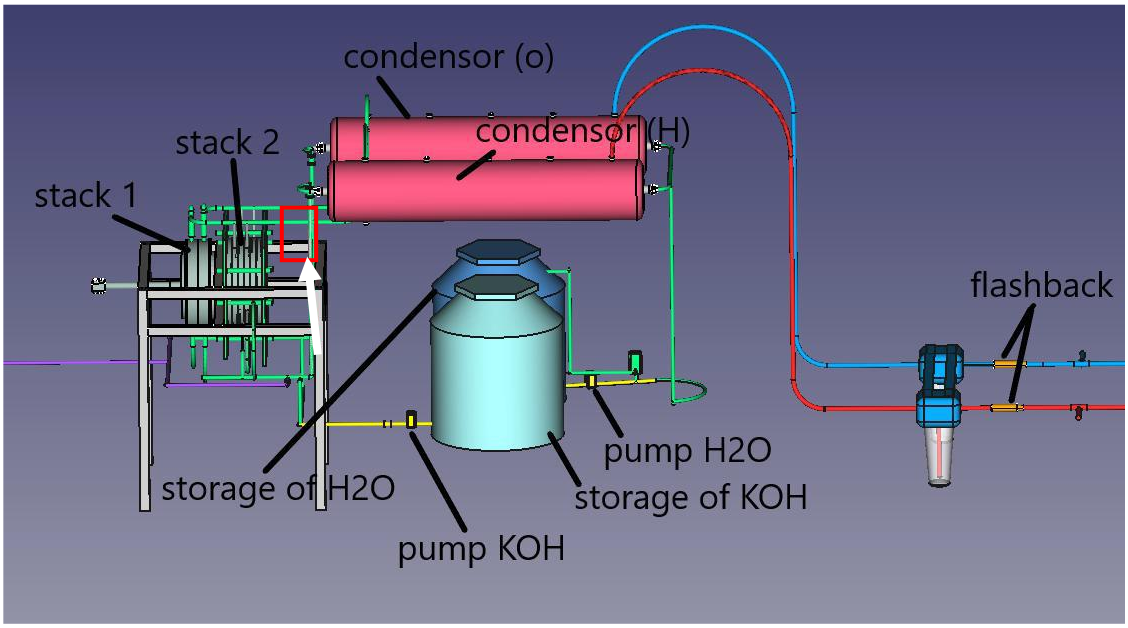
2.2. Gathering

In this part, all missed equipment was put in its place in the whole system.

2.2.1. Automatic valves

2.2.1.1. Gas Condenser Water Discharge

The Automatic valve that has to discharge water from condenser (H) will take place in the red rectangle, as shown in figure 4. The same thing for condenser (O).



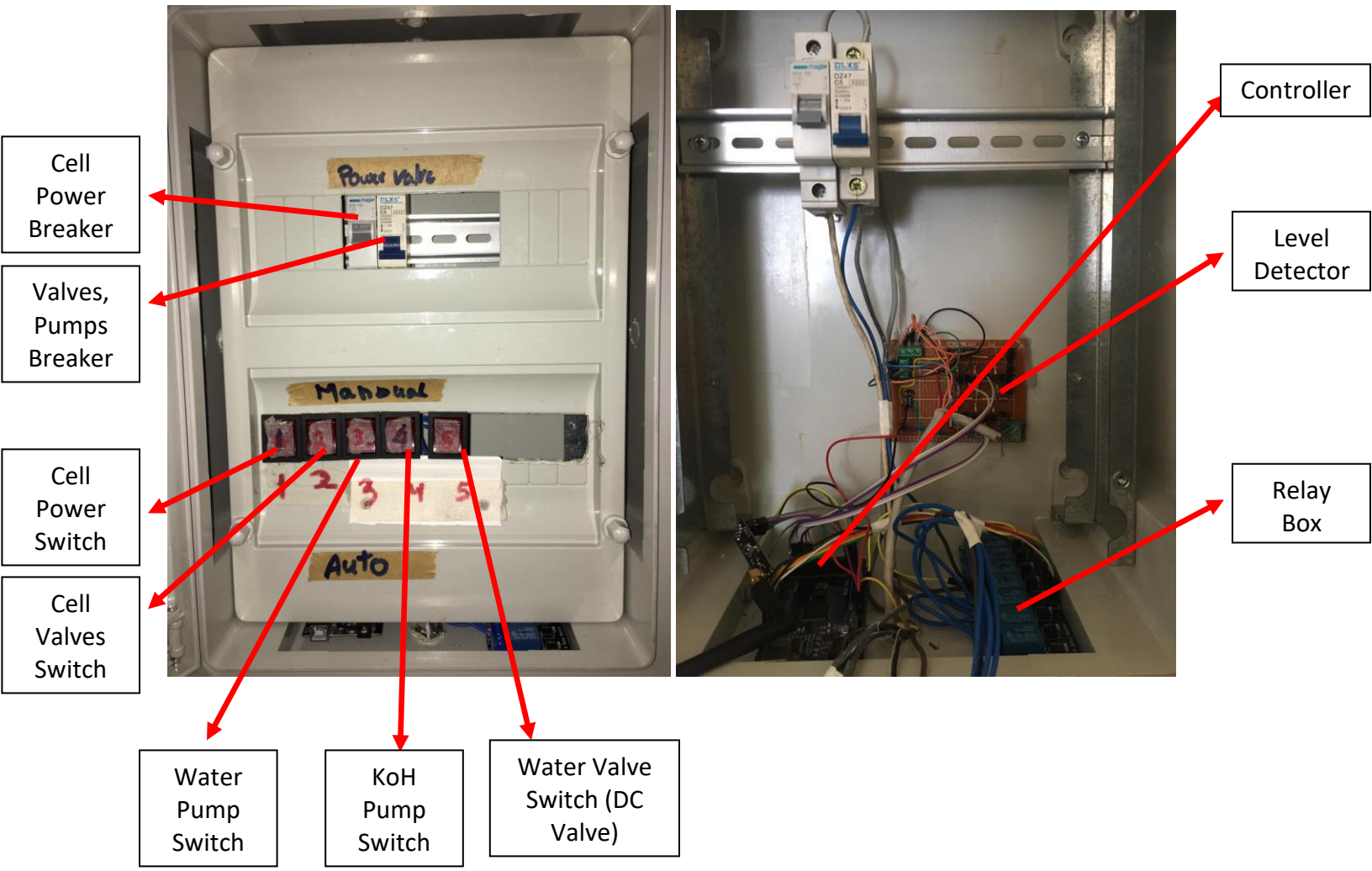
In order to connect Automatic valve to system, two M/M joint of hole size: 15 mm are needed, one for solenoid inlet, and one for its outlet. Figure 2 and 3, below, presents Automatic valve, and M/M joint.

Figure 5. M/M Joint

Figure 7. Valves with/without Joints

Figure 7, and 8 show how these two Automatic valves was gathered with the whole system.

3. Power Control Box



- Cell Power Breaker: Turns on/off the electrolyzer cell power source (12V @ 145A) and is rated at 10A.
- Valves and Pumps Breaker: Turns on/off all the system's electrical actuators. Rated at 5A.

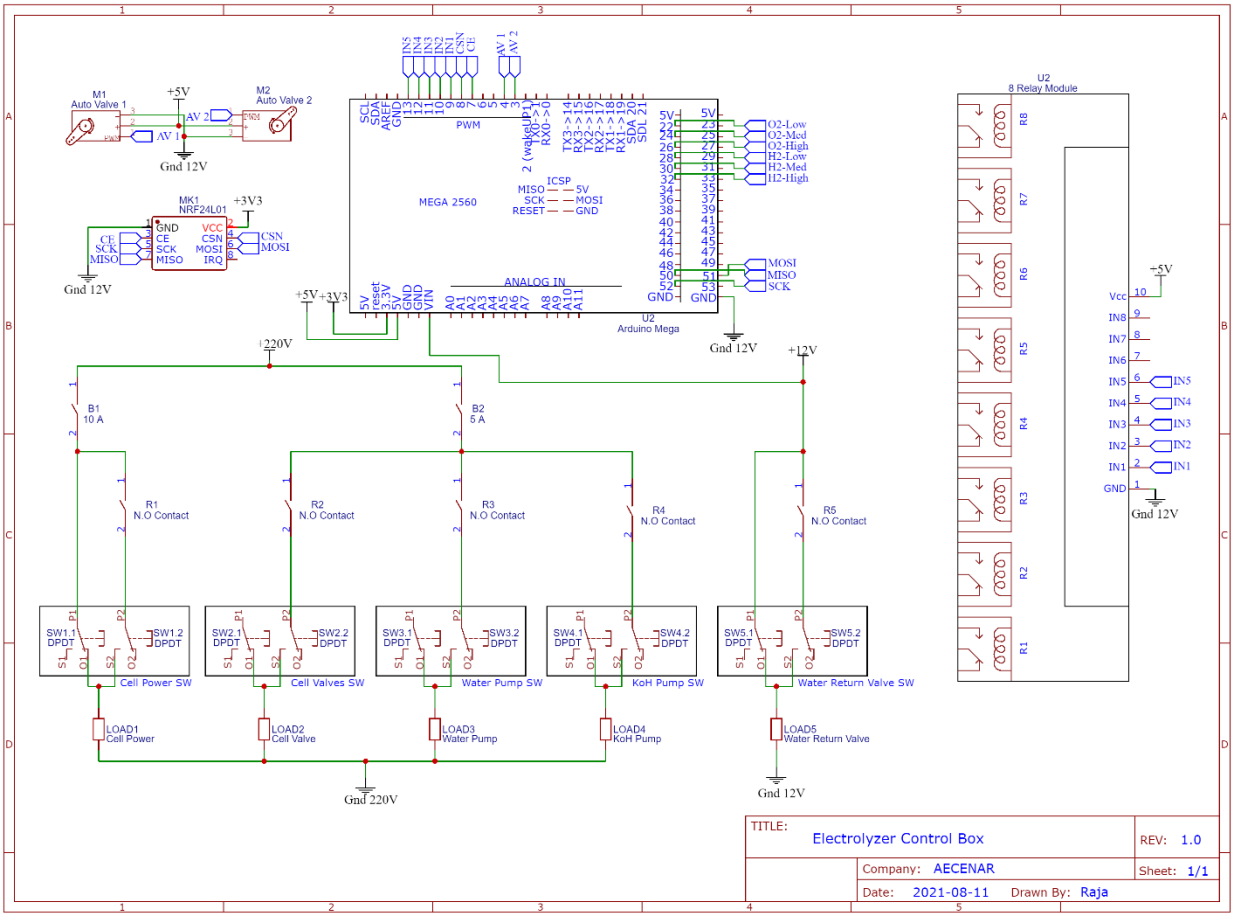
- Switch 1 (Cell Power): Manual at upper position in which the cell power source will turn ON directly if the breaker is ON and the switch is up. Otherwise, it will turn ON/OFF according to controller command if switched to lower position.
- Switch 2 (Cell Valves): Manual at upper position in which both cell valves will turn ON directly if the “valves and pumps” breaker is ON and the switch is in its upper position. Otherwise, it will turn ON/OFF according to controller command if switched to lower position.
- Switch 3 (Water Pump): Manual at upper position in which water pump will turn ON directly if the “valves and pumps” breaker is ON and the switch is in its upper position. Otherwise, it will turn ON/OFF according to controller command if switched to lower position.
- Switch 4 (KoH Pump): Manual at upper position in which KoH pump will turn ON directly if the “valves and pumps” breaker is ON and the switch is in its upper position. Otherwise, it will turn ON/OFF according to controller command if switched to lower position.
- Switch 5 (Water Valve): Manual at upper position in which water pump will turn ON directly if the 12 V power adapter is ON and the switch is in its upper position. Otherwise, it will turn ON/OFF according to controller command if switched to lower position.
- Controller: The controller used is Arduino Mega. A nrf24l01 radio frequency antenna is used to establish wireless communication between the plant and the remote station. It is used to control the relay box according to operator’s command (from the user-end GUI) and send back the KoH and water levels (Low – Medium – High).
- Level Detector: A circuit that detects the current liquid level (Low – Medium – High).
- Relay Box: Controlled contacts (8 channels).

3.1. Wiring Diagram

The wiring diagram should follow the safety standards and appropriate cable sizing to ensure maximum operating efficiency. The size of wires was selected upon the load required power. The below table shows the selected cable sizes with respect to every load consumption.

Load	Current Consumption (A)	Wire Size (mm^2)
Cell Power Source	~10	2.5
Cell Valves	0.145	1.5
Water Pump	Type equation here.	1.5
KoH Pump	Type equation here.	1.5
Water Return Valve (DC)	Type equation here.	1
Automatic Valves (DC)	0.35	1

The below figure illustrates the power system connections. The relay box is controlled through Arduino's digital pins according to a command received (by nrf24l01 module) from the user GUI. The Switches (SW1 to SW5) select the mod of operation. The Controller is responsible for sending back the O_2 and H_2 liquid levels to the user GUI through its transceiver module (nrf24l01). **NOTE: SW1 is MALFUNCTIONED and is not used at the moment!**

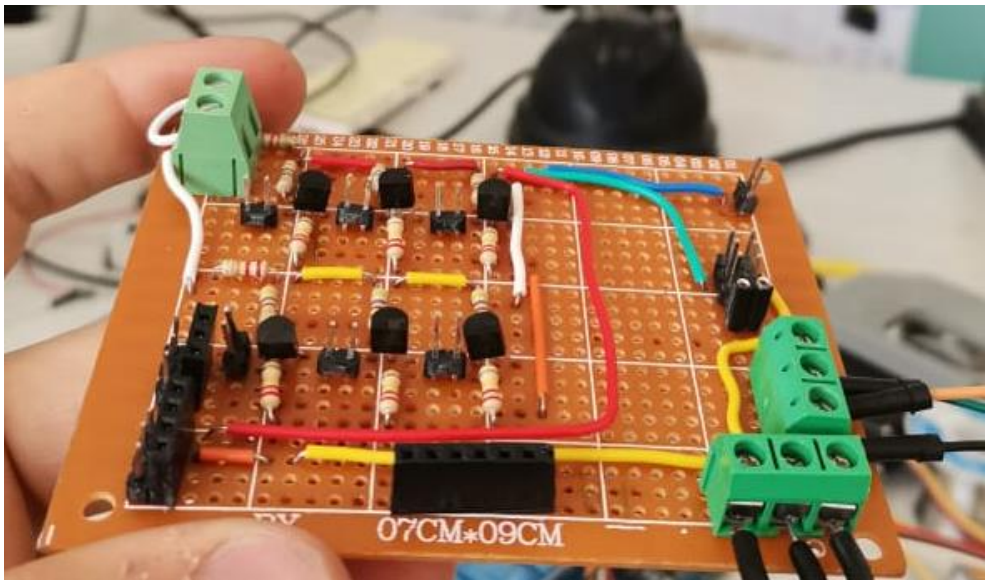


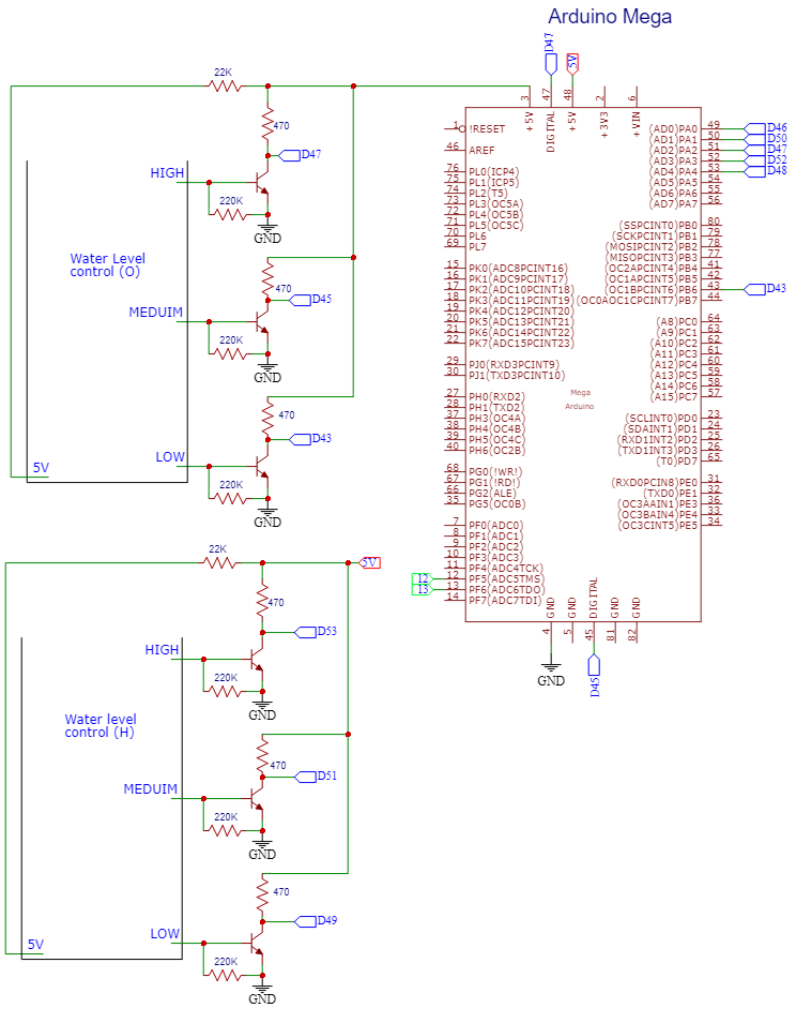
3.2. Level Detector

The level detector is an electronic circuit designed to monitor the current available amount of O_2 and H_2 levels in the containers. The circuit outputs 6 digital data signals

representing the current liquid level (3 for each container: Low – Med – High) compatible with the Arduino Controller (@ 5V).

3.2.1. Schematics and Principle of Operation

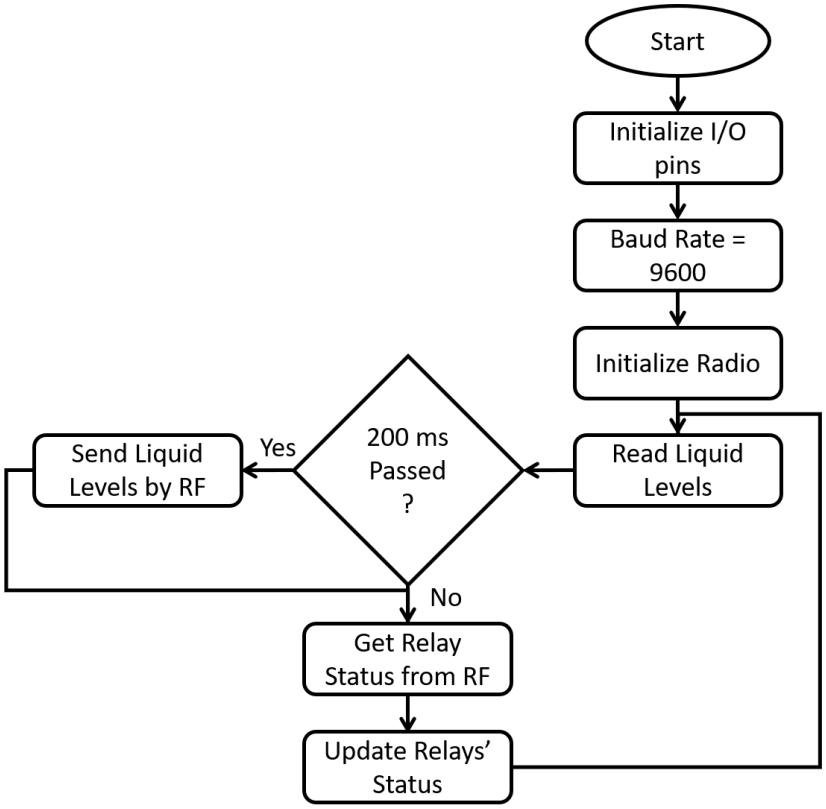




The above 2 figures illustrate the soldered board and the circuit schematic respectively. As can be seen from the schematic, each container includes 4 wires with different lengths and 3 transistor switches (each represents a certain state; Low – Med – High) whereas the 4th wire carries the positive 5V from the Arduino. The collectors of the transistors are connected to the Arduino's digital pins as input which are naturally pulled HIGH (5V). When liquid is present in the container, a short circuit is established between the 5V wire and the level wires (Low – Med – High) which in turn will bias the transistor and switch it ON. When a certain transistor is ON, the specific Arduino pin is pulled LOW (0 V). The below code is used to test the liquid level detector circuit.

liquidLevel.ino

3.3. Flow Chart and Source Code



The controller of the electrolyzer plant will send the O_2 and H_2 levels to the user GUI every 200 ms. The RF module (nrf24l01) will switch its role every 200 ms from receiving to sending. NOTE: The below 2 codes should be in the same folder in order for them to work.

Radio.ino

Station_Send_Receive.ino

The source code (text) is shown below:

```
/*FOR ARDUINO MEGA: MASTER ADDRESS*/

#include <SPI.h>
#include <Wire.h>
#include <nRF24L01.h>
```

```
#include <RF24.h>
#include<Servo.h>

#define cePin 3
#define csnPin 4

RF24 radio(cePin, csnPin); // Create a Radio
//Servo av_oxygen, av_hydrogen;

// RADIO VARIABLES //
const byte slaveAddress[5] = {'R', 'x', 'A', 'A', 'A'};
const byte masterAddress[5] = {'T', 'X', 'a', 'a', 'a'};
char dataToSend[10] = "Message 0";
char txNum = '0';
int dataReceived[2]; // to hold the data from the slave - must match
replyData[] in the slave
bool newData = false;

unsigned long currentMillis;
unsigned long prevMillis;
unsigned long txIntervalMillis = 200; // send once per 200 ms

// RELAY PINS //
byte relay_pins[5] = {13, 12, 11, 10, 9};
int tx_failed_count = 0;
boolean relay_status[5] = {0, 0, 0, 0, 0};

// LIQUID LEVEL //
byte oxygenLevel_pins[3] = {22, 24, 26}; // LOW - MED - HIGH
byte hydrogenLevel_pins[3] = {28, 30, 32}; // LOW - MED - HIGH
byte water_hyd_level[6];
void setup() {
  Serial.begin(9600);

  radio_init();

  for (int i = relay_pins[0] ; i > 8; i--) pinMode(i, OUTPUT);
  for (int i = oxygenLevel_pins[0]; i < hydrogenLevel_pins[2] + 1; i
+= 2) pinMode(i, INPUT);
  for (int i = 0; i < 4; i++) digitalWrite(relay_pins[i], 1);
}

void loop() {
  for (int i = 0; i < 3; i++) {
    water_hyd_level[i] = 1; //!digitalRead(oxygenLevel_pins[i])
    water_hyd_level[i + 3] = 0;//!digitalRead(hydrogenLevel_pins[i]);
  }

  currentMillis = millis();
  if (currentMillis - prevMillis >= txIntervalMillis) {
    send();
    prevMillis = millis();
```

```
  }
  getData();
  showData();

  for (int i = 0; i < 5; i++)
    digitalWrite(relay_pins[i], !relay_status[i]);
}
void radio_init() {
  radio.begin();
  radio.setDataRate( RF24_250KBPS );

  radio.openWritingPipe(slaveAddress);
  radio.openReadingPipe(1, masterAddress);

  radio.setRetries(3, 5); // delay, count
  send(); // to get things started
  prevMillis = millis(); // set clock
}

// Will send Liquid level (H2O - Hyd) levels to the station @ 500 ms
void send() {

  radio.stopListening();
  bool rslt;
  rslt = radio.write( &water_hyd_level, sizeof(water_hyd_level) );
  radio.startListening();

  Serial.print("Data Sent (Levels) ");
  for (int i = 0; i < 6; i++)
    Serial.print(String(water_hyd_level[i]) + " ");
  Serial.println();

  if (rslt) {
    Serial.println("  Acknowledge received");
    updateMessage();
  }
  else {
    Serial.println("  Tx failed");
    tx_failed_count++;

    // If the transmission failed for 10 times
    if (tx_failed_count == 10) {
      // Turn OFF ALL Relays
      for (int i = 0; i < 5; i++)
        relay_status[i] = 0;

      tx_failed_count = 0;
    }
  }
  Serial.println();
}
```

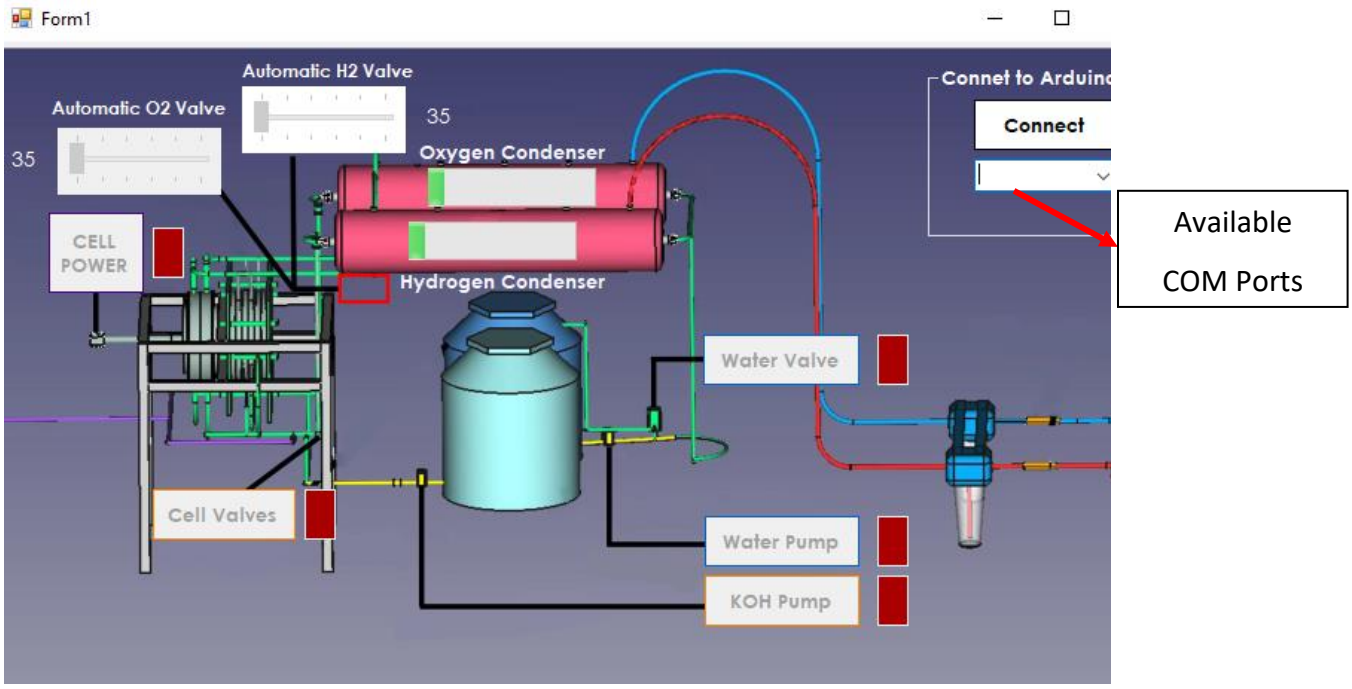
```
// Will receive relay status from the station @ 500 ms
void getData() {

  if ( radio.available() ) {
    radio.read( &relay_status, sizeof(relay_status) );
    newData = true;
  }
}

void showData() {
  if (newData == true) {
    Serial.print("Data received Relay Status ");
    for (int i = 0; i < 4; i++)
      Serial.print(String(relay_status[i]) + " ");
    Serial.println();
    newData = false;
  }
}

// JUST FOR TESTS! IS NOT PART OF THE CODE
void updateMessage() {
  // so you can see that new data is being sent
  txNum += 1;
  if (txNum > '9') {
    txNum = '0';
  }
  dataToSend[8] = txNum;
}
```

4. User-End

This section discusses the interaction between the user GUI and the Arduino Uno controller which is used to receive relay status data from the GUI and send them to the electrolyzer controller.

4.1. GUI

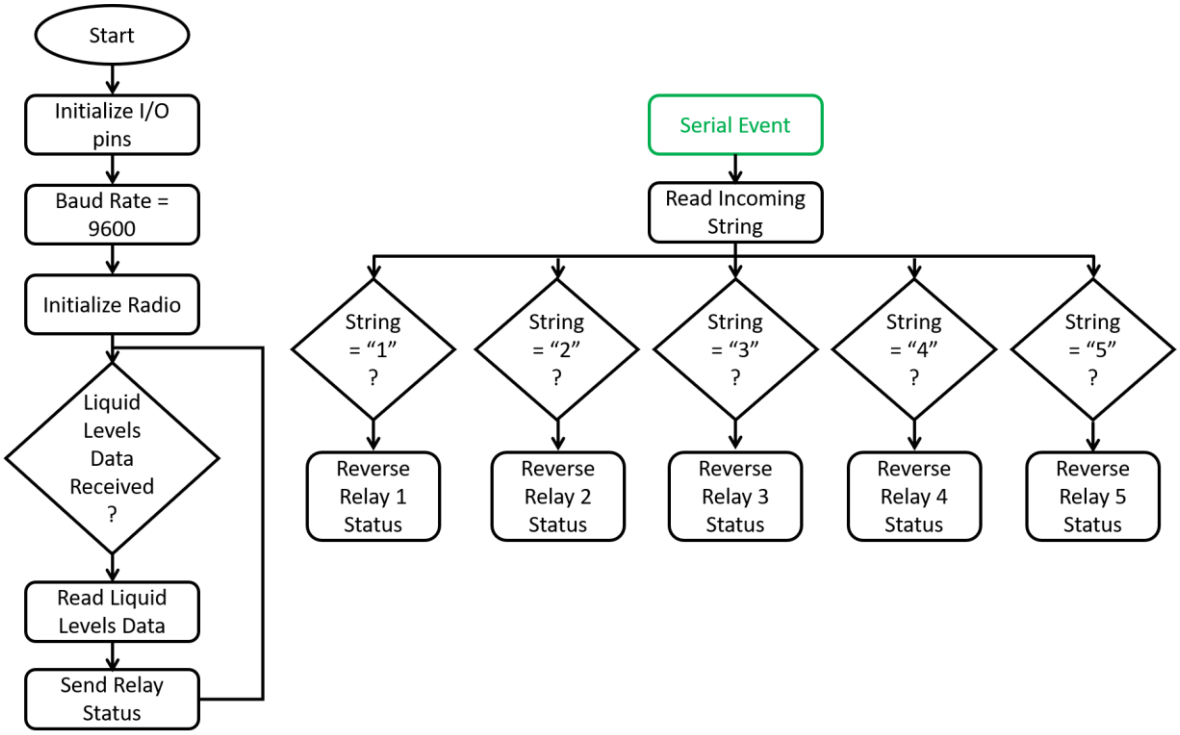
The GUI interface is shown in the image below. It is divided into 3 sections; The Power and connect, monitoring, and control.



- Connect Button: This button is used to connect to the Arduino controller. It uses the COM port shown in the COM port combo box below it. All buttons will not work until the computer is connected to the station Arduino.
- Cell Power Button: if the electrolyzer is set to Auto mode and this button is pressed, the cell power (welding machine) is activated and its indicator (red box next to it) should turn green.
- The H_2 and O_2 Bars: Will indicate the current amount of liquid found in the condenser.
- Automatic valves (1 and 2): These sliders will control the servo valves at the indicated angle label (number to the left and right).
- Water Pump, KoH Pump, Water Return Valves, and Cell Valves: These buttons will control the respective appliances.

Elec GUI.zip

4.2. Controller



When Data come from the GUI, the "Serial Event" event handler function is stimulated. This function will read the incoming string sent from the GUI and will take the decision accordingly. The codes of the user station are attached below. NOTE: The below 2 codes should be in the same folder in order for them to work.

The Arduino code (in Text) is shown below:

```
/*FOR ARDUINO UNO: SLAVE ADDRESS*/

#include <SPI.h>
#include <Wire.h>
#include <nRF24L01.h>
#include <RF24.h>

#define cePin 7
#define csnPin 8

RF24 radio(cePin, csnPin); // Create a Radio
```

```
// FOR GUI CONFIG //
String inputString = "";         // a string to hold incoming data
boolean stringComplete = false, ledState = false;  // whether the
string is complete
String commandString = "";
boolean isStarted = false;
boolean dataToSend[11];

int autoValve;

// RADIO VARIABLES //
const byte slaveAddress[5] = {'R', 'x', 'A', 'A', 'A'};
const byte masterAddress[5] = {'T', 'X', 'a', 'a', 'a'};

bool newData = false, txDone;

unsigned long currentMillis;
unsigned long prevMillis;
unsigned long txIntervalMillis = 200; // send once per 200 ms

// CONTROL ACTIONS //
boolean oxy_hyd_levels[6] = {1, 0, 0, 1, 0, 0};
boolean relay_status[5] = {0, 0, 0, 0, 0};

int oxy_hyd_levels_size = sizeof(oxy_hyd_levels) /
sizeof(oxy_hyd_levels[0]);
int relayStatus_size = sizeof(relay_status) / sizeof(relay_status[0]);
const int dataToSend_size = oxy_hyd_levels_size + relayStatus_size;

void setup() {

  Serial.begin(9600);

  radio_init();

}

void loop() {
  // Data Received From GUI
  if (stringComplete)  {
    stringComplete = false;

    int commandLength = commandString.length();
    commandString.remove(0, 1);
    commandString.remove(commandLength - 1);

    if (commandString.equals("STAR")) isStarted = true;
    else if (commandString.equals("STOP")) isStarted = false;
    else if (commandString.equals("1")) relay_status[0] =
!relay_status[0];
```

```
    else if (commandString.equals("2")) relay_status[1] =
!relay_status[1];
    else if (commandString.equals("3")) relay_status[2] =
!relay_status[2];
    else if (commandString.equals("4")) relay_status[3] =
!relay_status[3];
    else if (commandString.equals("5")) relay_status[4] =
!relay_status[4];
    else autoValve = commandString.toInt();
  }

  // Data to send back to GUI
  if (isStarted) {
    String inString = "";

    for (int i = 0; i < dataToSend_size; i++)  {
      String sep = ",";
      if (i == dataToSend_size - 1) sep = "";

      if (i < relayStatus_size) dataToSend[i] = relay_status[i];
      else dataToSend[i] = oxy_hyd_levels[i - relayStatus_size];

      inString += String(dataToSend[i]);
      //dataToSend = {R1, R2, R3, R4, R5, O2-L, O2-M, O2-H, H2-L, H2-
M, H2-H}
      //Serial.println(String(dataToSend[i]) + sep);
    }
    Serial.print(inString+ "\n");
  }

  // Control of Auto Valves 1 and 2
  if (autoValve > 349 && autoValve < 1351) {
    // Code for auto valve 1
    int autoValve1_angle = autoValve / 10;

  }
  else if (autoValve > 3000) {
    // Code for auto valve 2
    int autoValve2_angle = autoValve / 100;
  }

  // RF FUNCTIONS //
  // Get the liquid levels from the plant
  getData();
  // Send the relay status to the plant
  send();

  delay(10);
}

void serialEvent() {
  while (Serial.available()) {
```

```
    commandString = Serial.readStringUntil('\n');
  }
  stringComplete = true;
}
void radio_init() {
  radio.begin();
  radio.setDataRate( RF24_250KBPS );

  radio.openWritingPipe(masterAddress); // NB these are swapped
compared to the master
  radio.openReadingPipe(1, slaveAddress);

  radio.setRetries(3, 5); // delay, count
  radio.startListening();
}

// Will send relay status to plant
void send() {
  if (newData == true) {
    radio.stopListening();
    bool rslt;
    rslt = radio.write(&relay_status, sizeof(relay_status) );
    radio.startListening();

    if (rslt) txDone = true;
    else txDone = false;

    newData = false;
  }
}

// Will work every txIntervalMillis @ Plant Side
void getData() {
  if ( radio.available() ) {
    radio.read( &oxy_hyd_levels, sizeof(oxy_hyd_levels) );
    newData = true;
  }
}
```