WEEK 12 PRESENTATION

LIQUID PROPULSION

Tasks Allocated for this Week

01 P&ID Diagram

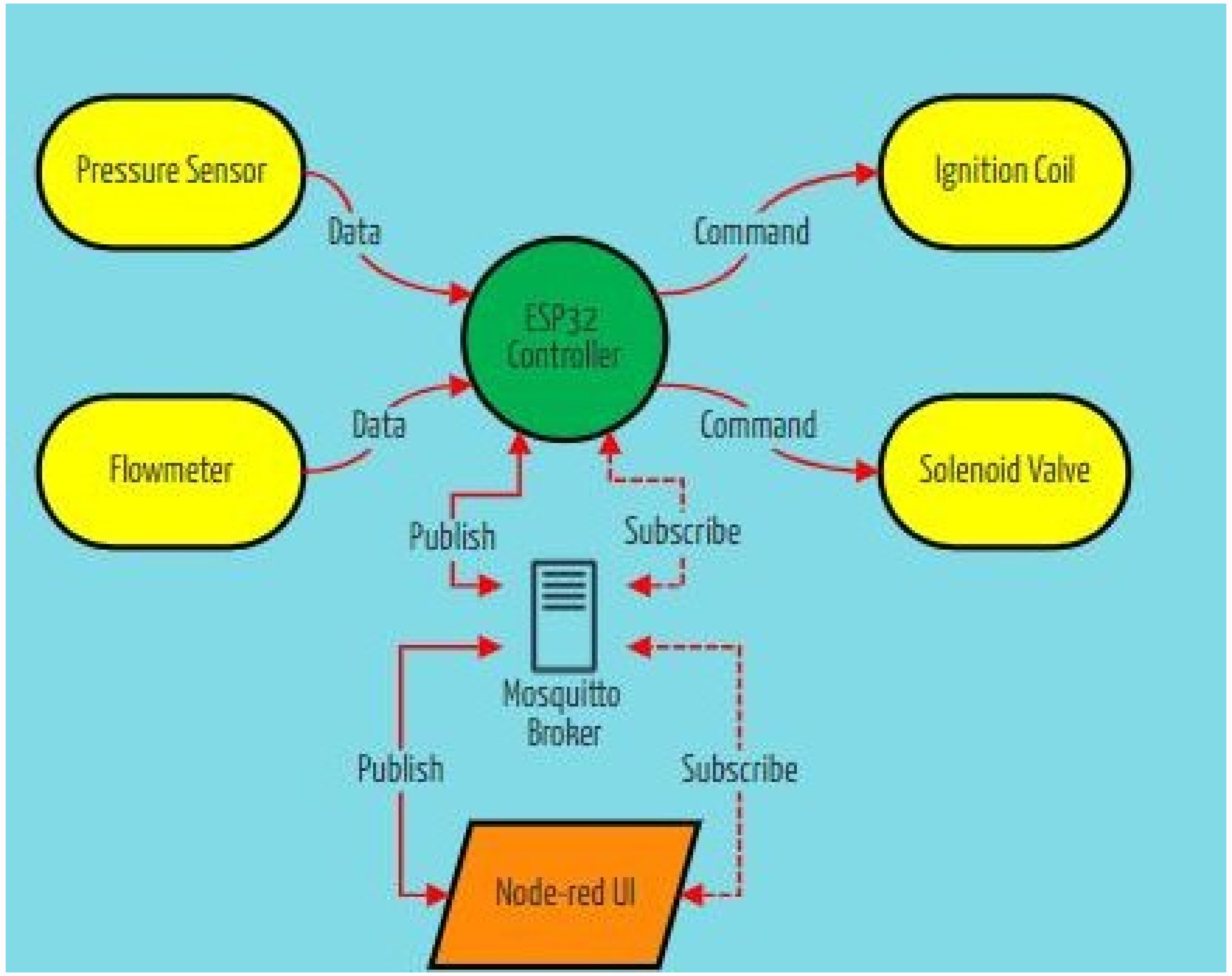
02 Remote control

03 Engine control system

04 Filling of tanks

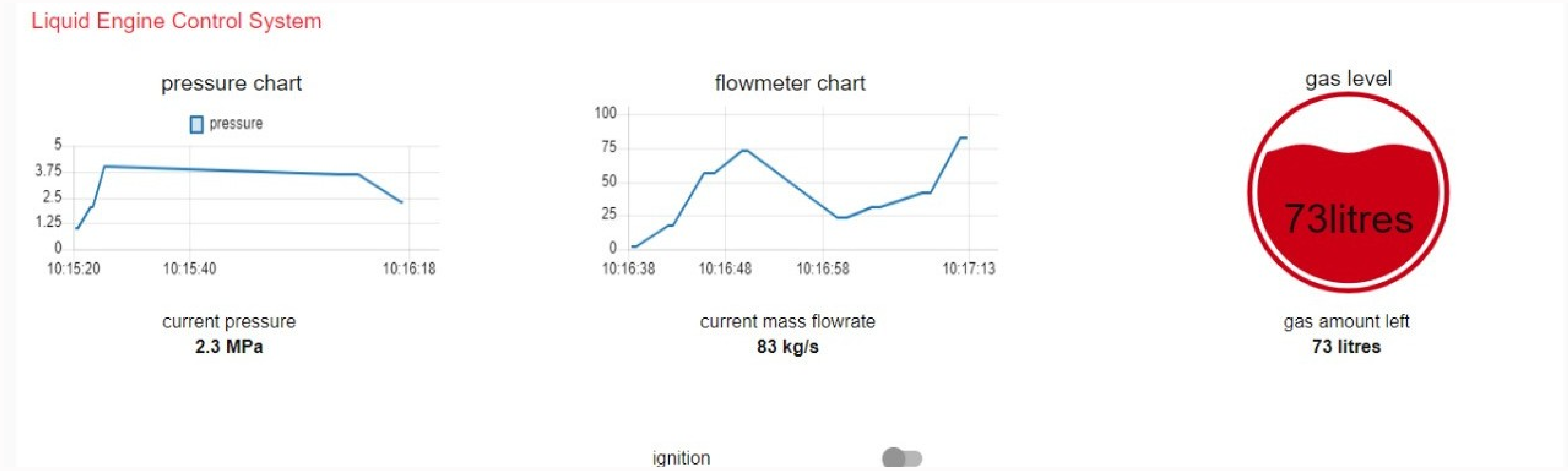
07

Remote control flowchart



- sensor sends data via serial communication.
- data is published then the dashboard subscribes it

Liquid engine control system

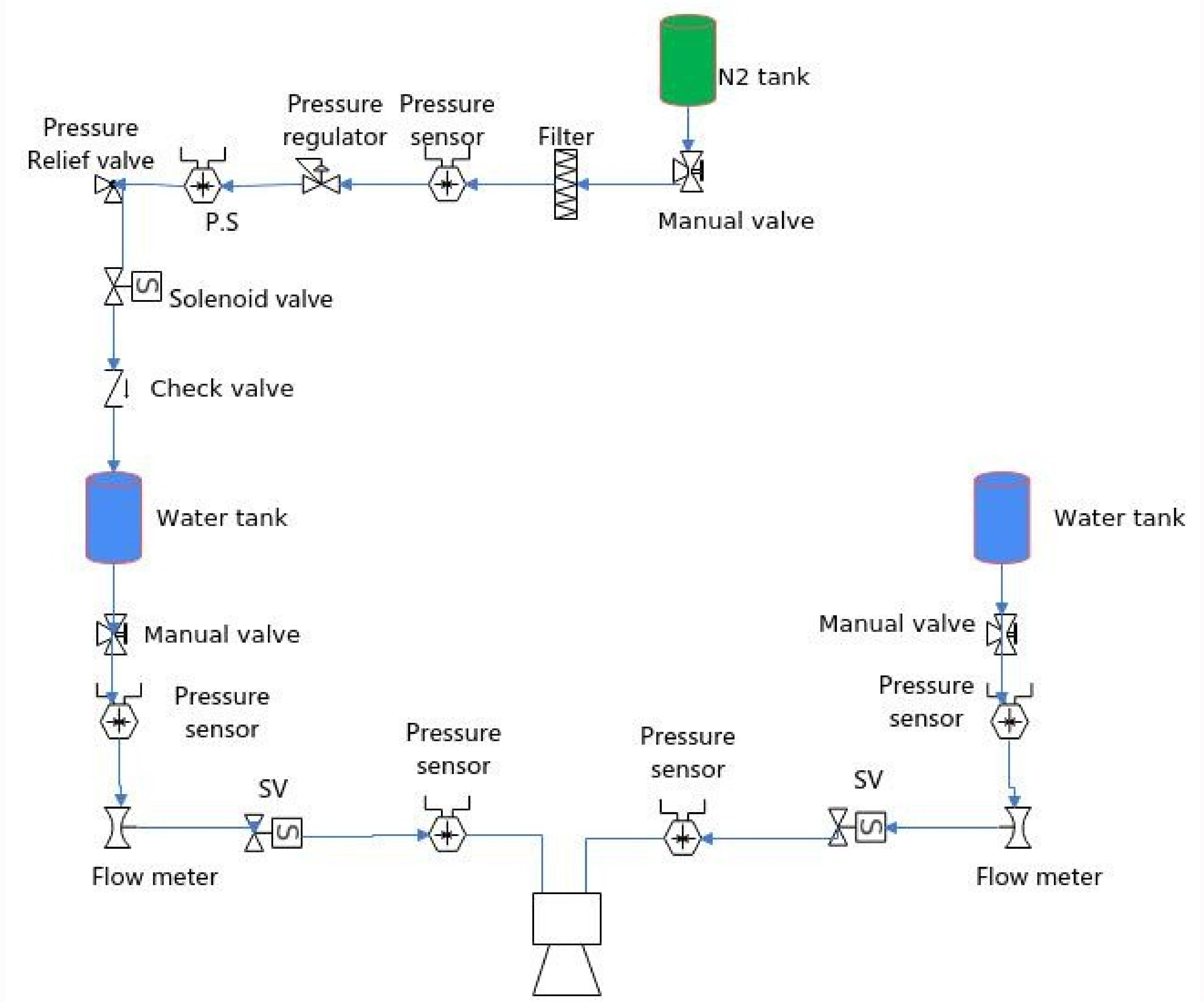


current pressure
2.3 MPa

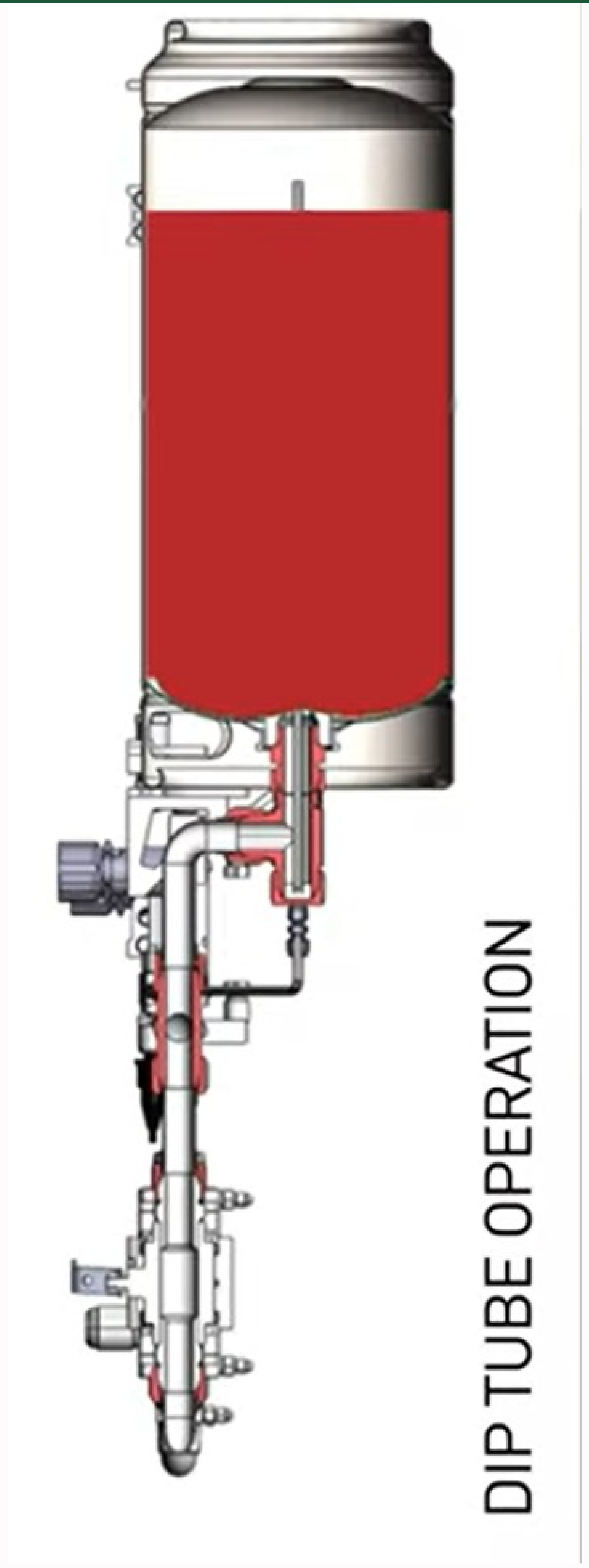
current mass flowrate
83 kg/s

gas amount left
73 litres

PID DIAGRAM


N2 tank
Pressure regulator
Pressure sensor
Filter
Pressure Relief valve
P.S
Manual valve
Solenoid valve
Check valve
Water tank
Water tank
Manual valve
Manual valve
Pressure sensor
Pressure sensor
Pressure sensor
Pressure sensor
SV
SV
Flow meter
Flow meter

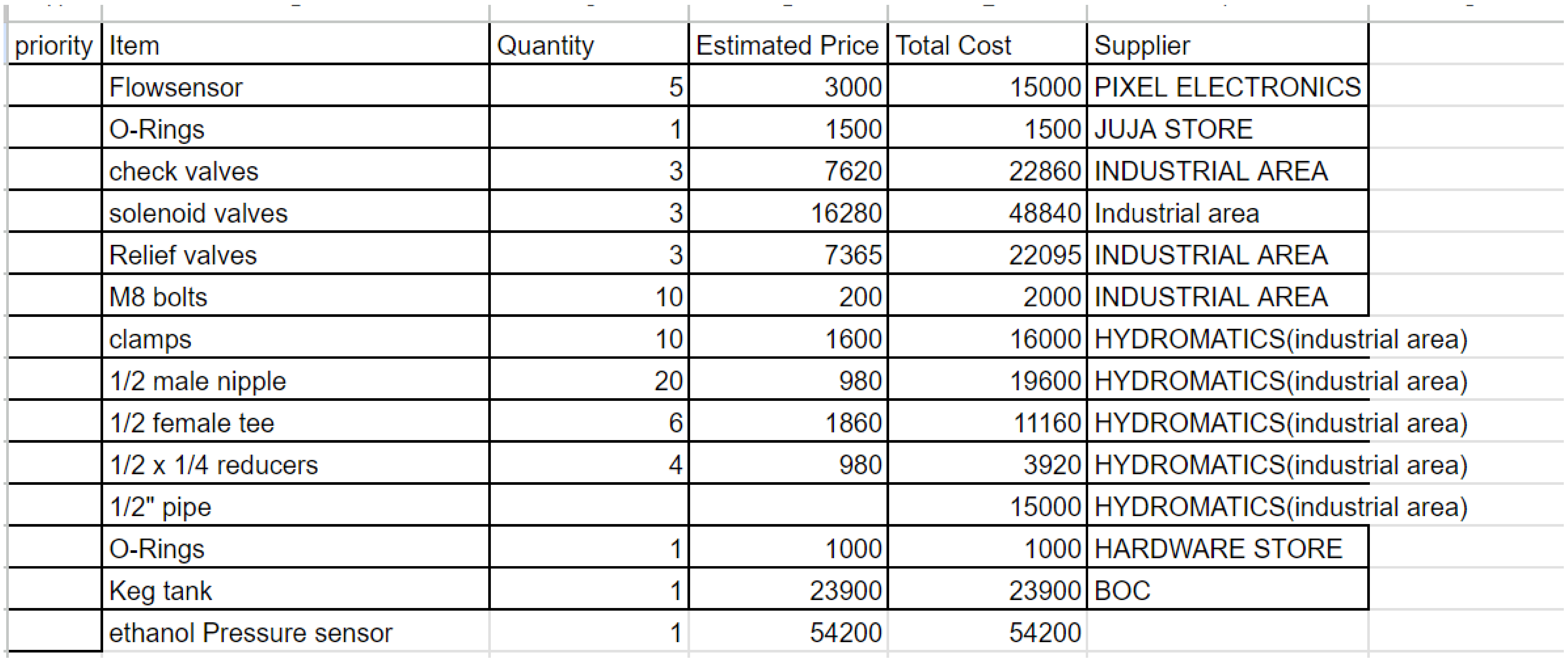
FILLING OF TANKS



-Method of fillin water into the tanks

-Using the dip tube operation

WATER TEST MATERIALS



priority	Item	Quantity	Estimated Price	Total Cost	Supplier
	Flowsensor	5	3000	15000	PIXEL ELECTRONICS
	O-Rings	1	1500	1500	JUJA STORE
	check valves	3	7620	22860	INDUSTRIAL AREA
	solenoid valves	3	16280	48840	Industrial area
	Relief valves	3	7365	22095	INDUSTRIAL AREA
	M8 bolts	10	200	2000	INDUSTRIAL AREA
	clamps	10	1600	16000	HYDROMATICS(industrial area)
	1/2 male nipple	20	980	19600	HYDROMATICS(industrial area)
	1/2 female tee	6	1860	11160	HYDROMATICS(industrial area)
	1/2 x 1/4 reducers	4	980	3920	HYDROMATICS(industrial area)
	1/2" pipe			15000	HYDROMATICS(industrial area)
	O-Rings	1	1000	1000	HARDWARE STORE
	Keg tank	1	23900	23900	BOC
	ethanol Pressure sensor	1	54200	54200	

Solid Communication Code Modularisation

The team decided to assist the Solid Team in revising the communication code.

Previous Code features -> works well, but trouble-shooting and adding features is a bit hard.

New Code features

- Borrows the flow of the previous code.
- Is Modular in nature
- Code is documented

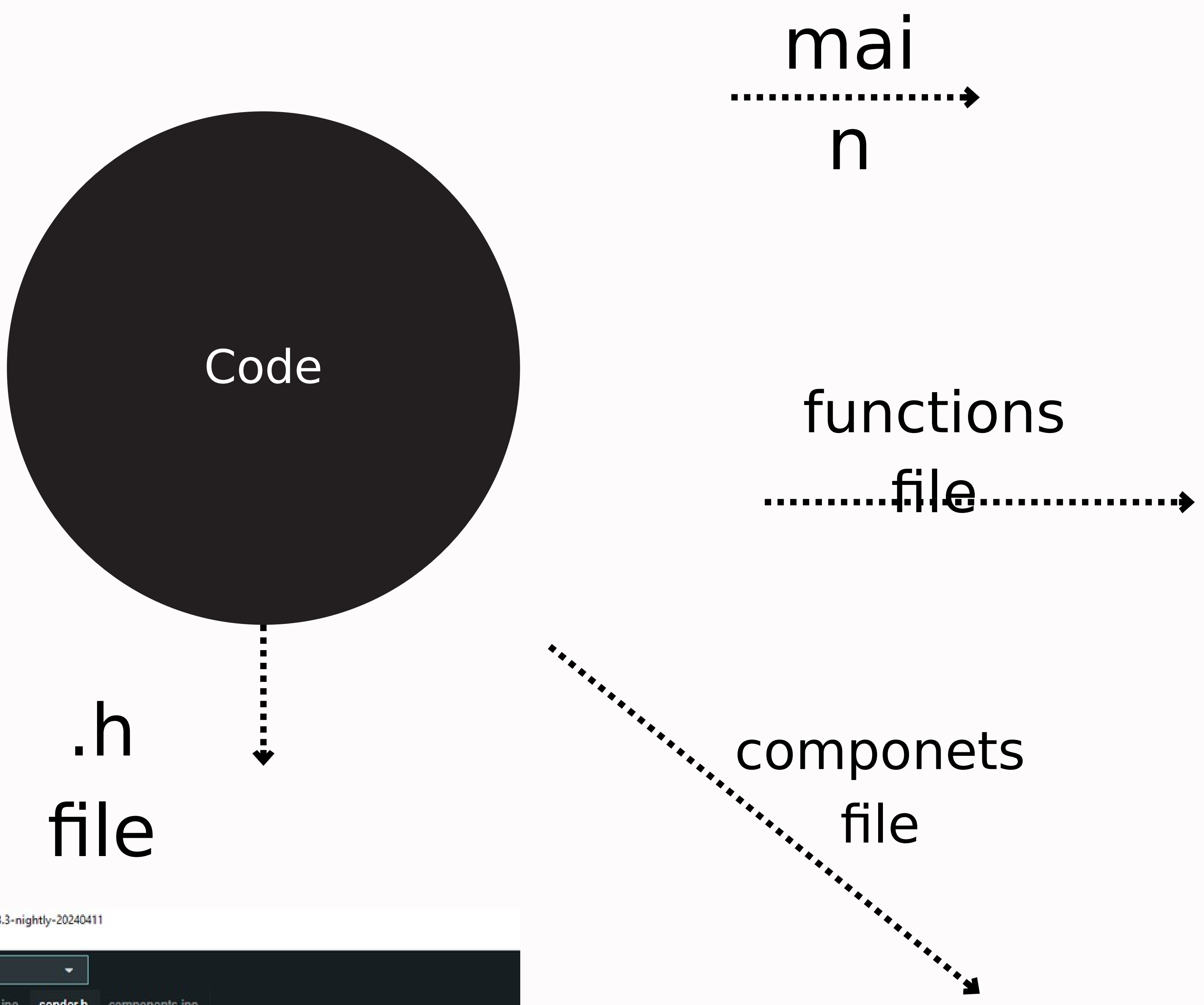
Benefits

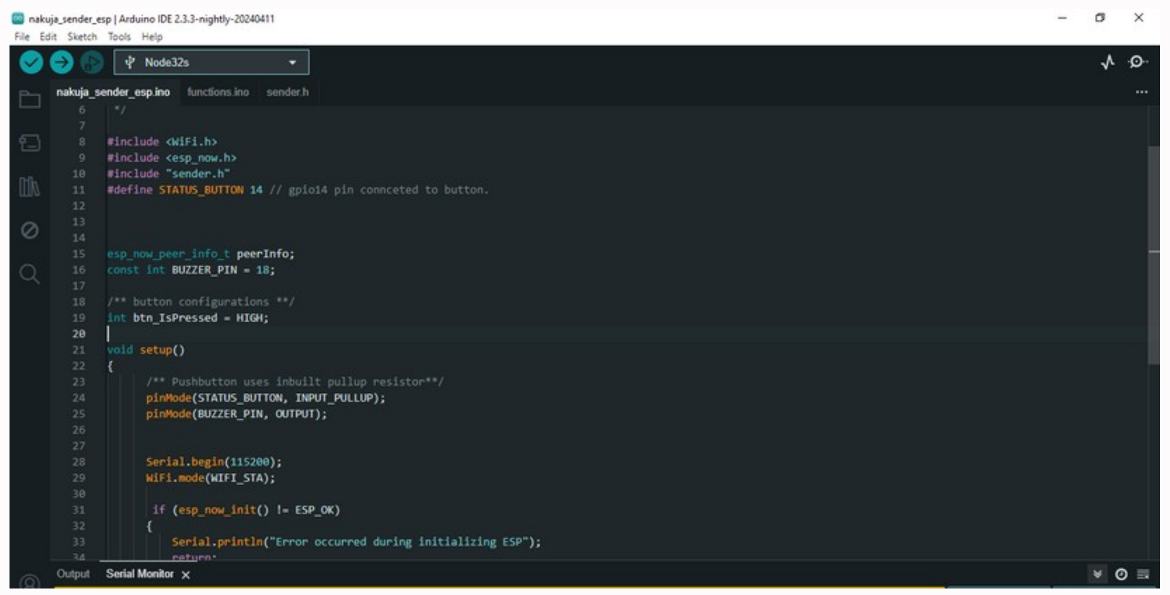
- Easy maintainance.
- Allows easy addition of addons

New Code Structure

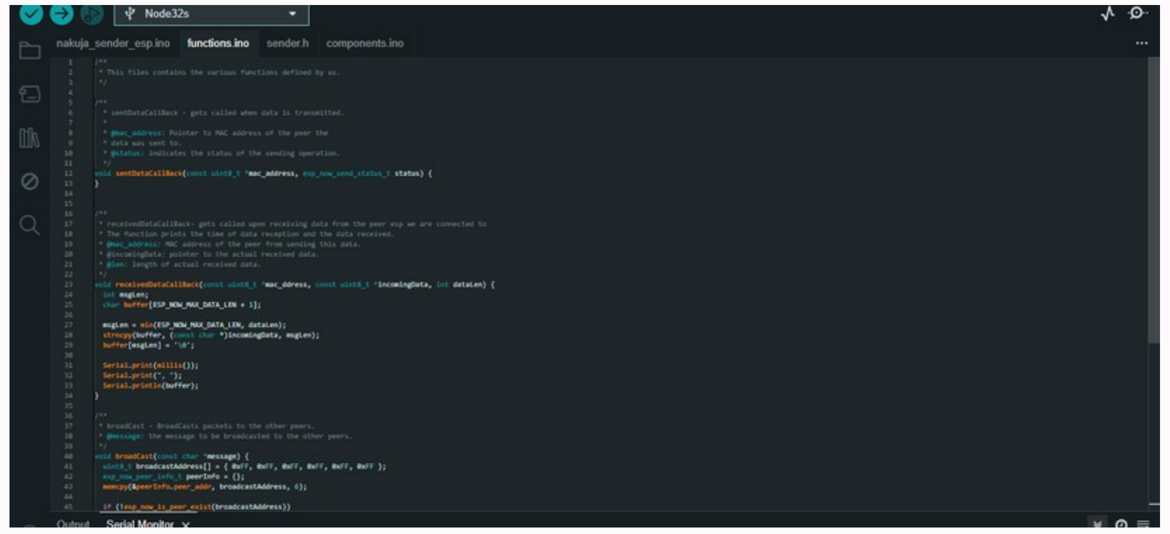
Code

- Main file (contains the setup() and loop()
- function file (contains helper functions eg callbacks ,broadca
  receiver functions
- components file (you can add and configure components
  i.e loadcell, LEDs ... here)
- .h files (enables the files in a project to work together

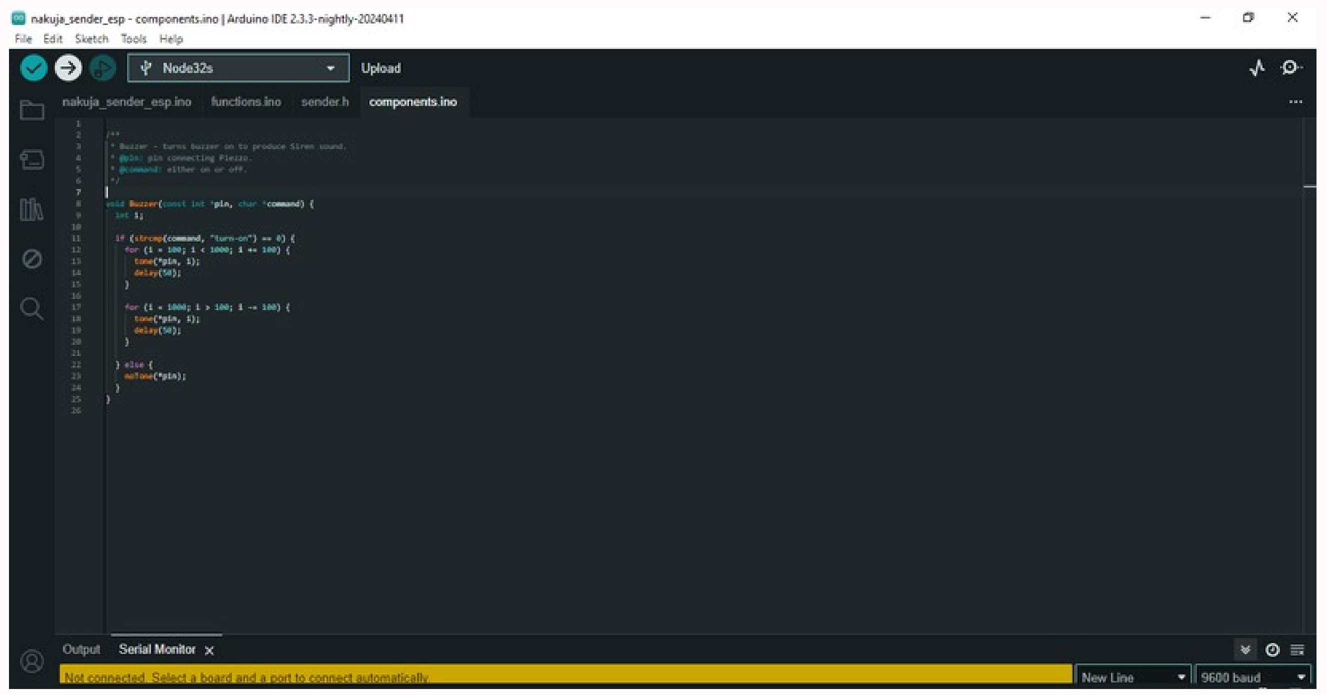



nakuja_sender_esp | Arduino IDE 2.3.3-nightly-20240411

```
 */

#include <WiFi.h>
#include <esp_now.h>
#include "sender.h"
#define STATUS_BUTTON 14 // gpio14 pin connceted to button.



esp_now_peer_info_t peerInfo;
const int BUZZER_PIN = 18;

/** button configurations **/
int btn_IsPressed = HIGH;

void setup()
{
    /** Pushbutton uses inbuilt pullup resistor**/
    pinMode(STATUS_BUTTON, INPUT_PULLUP);
    pinMode(BUZZER_PIN, OUTPUT);


    Serial.begin(115200);
    WiFi.mode(WIFI_STA);

     if (esp_now_init() != ESP_OK)
    {
      Serial.println("Error occurred during initializing ESP");
      return;
```

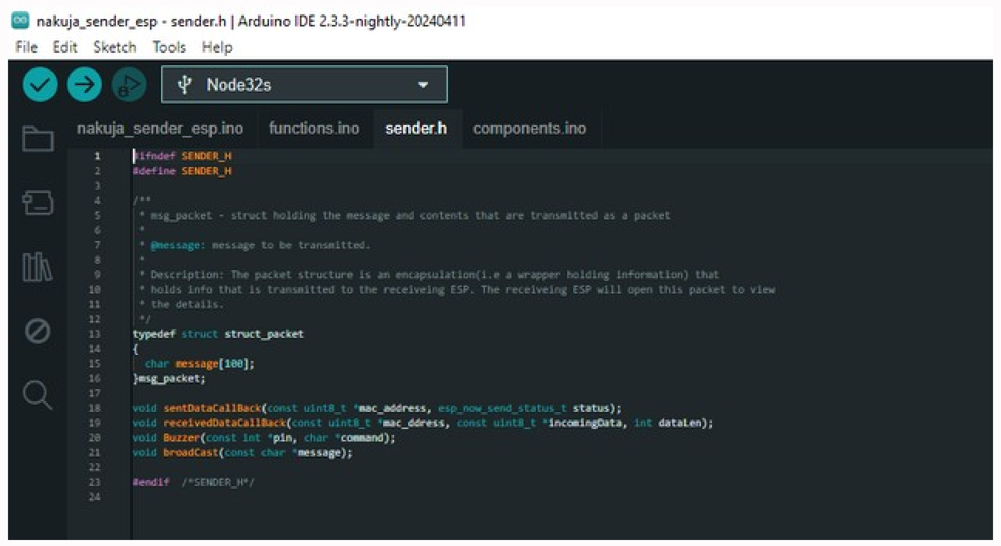


```
/**
 * This files contains the various functions defined by us.
 */

/**
 * sentDataCallBack - gets called when data is transmitted.
 *
 * @mac_address: Pointer to MAC address of the peer the
 * data was sent to.
 * @status: indicates the status of the sending operation.
 */
void sentDataCallBack(const uint8_t *mac_address, esp_now_send_status_t status) {
}


/**
 * receivedDataCallBack- gets called upon receiving data from the peer esp we are connected to
 * The function prints the time of data reception and the data received.
 * @mac_address: MAC address of the peer from sending this data.
 * @incomingData: pointer to the actual received data.
 * @len: length of actual received data.
 */
void receivedDataCallBack(const uint8_t *mac_ddress, const uint8_t *incomingData, int dataLen) {
  int msgLen;
  char buffer[ESP_NOW_MAX_DATA_LEN + 1];

  msgLen = min(ESP_NOW_MAX_DATA_LEN, dataLen);
  strncpy(buffer, (const char *)incomingData, msgLen);
  buffer[msgLen] = '\0';

  Serial.print(millis());
  Serial.print(", ");
  Serial.println(buffer);
}

/**
 * broadCast - BroadCasts packets to the other peers.
 * @message: the message to be broadcasted to the other peers.
 */
void broadCast(const char *message) {
  uint8_t broadcastAddress[] = { 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF };
  esp_now_peer_info_t peerInfo = {};
  memcpy(&peerInfo.peer_addr, broadcastAddress, 6);

  if (!esp_now_is_peer_exist(broadcastAddress))
```


nakuja_sender_esp - components.ino | Arduino IDE 2.3.3-nightly-20240411

```

/**
 * Buzzer - turns buzzer on to produce Siren sound.
 * @pin: pin connecting Piezzo.
 * @command: either on or off.
 */

void Buzzer(const int *pin, char *command) {
  int i;

  if (strcmp(command, "turn-on") == 0) {
    for (i = 100; i < 1000; i += 100) {
      tone(*pin, i);
      delay(50);
    }

    for (i = 1000; i > 100; i -= 100) {
      tone(*pin, i);
      delay(50);
    }

  } else {
    noTone(*pin);
  }
}
```

Not connected. Select a board and a port to connect automatically.


nakuja_sender_esp - sender.h | Arduino IDE 2.3.3-nightly-20240411

```
#ifndef SENDER_H
#define SENDER_H

/**
 * msg_packet - struct holding the message and contents that are transmitted as a packet
 *
 * @message: message to be transmitted.
 *
 * Description: The packet structure is an encapsulation(i.e a wrapper holding information) that
 * holds info that is transmitted to the receiveing ESP. The receiveing ESP will open this packet to view
 * the details.
 */
typedef struct struct_packet
{
  char message[100];
}msg_packet;

void sentDataCallBack(const uint8_t *mac_address, esp_now_send_status_t status);
void receivedDataCallBack(const uint8_t *mac_ddress, const uint8_t *incomingData, int dataLen);
void Buzzer(const int *pin, char *command);
void broadCast(const char *message);

#endif  /*SENDER_H*/

```

WEEK 13 OBJECTIVES

CONNECT ESP32s TO MOSQUITO SERVERS

ADDING BUZZER AND LED LIGHTS

RESEARCH ON PUMP-FED/PRESSURE FED SYSTEM

THANK YOU