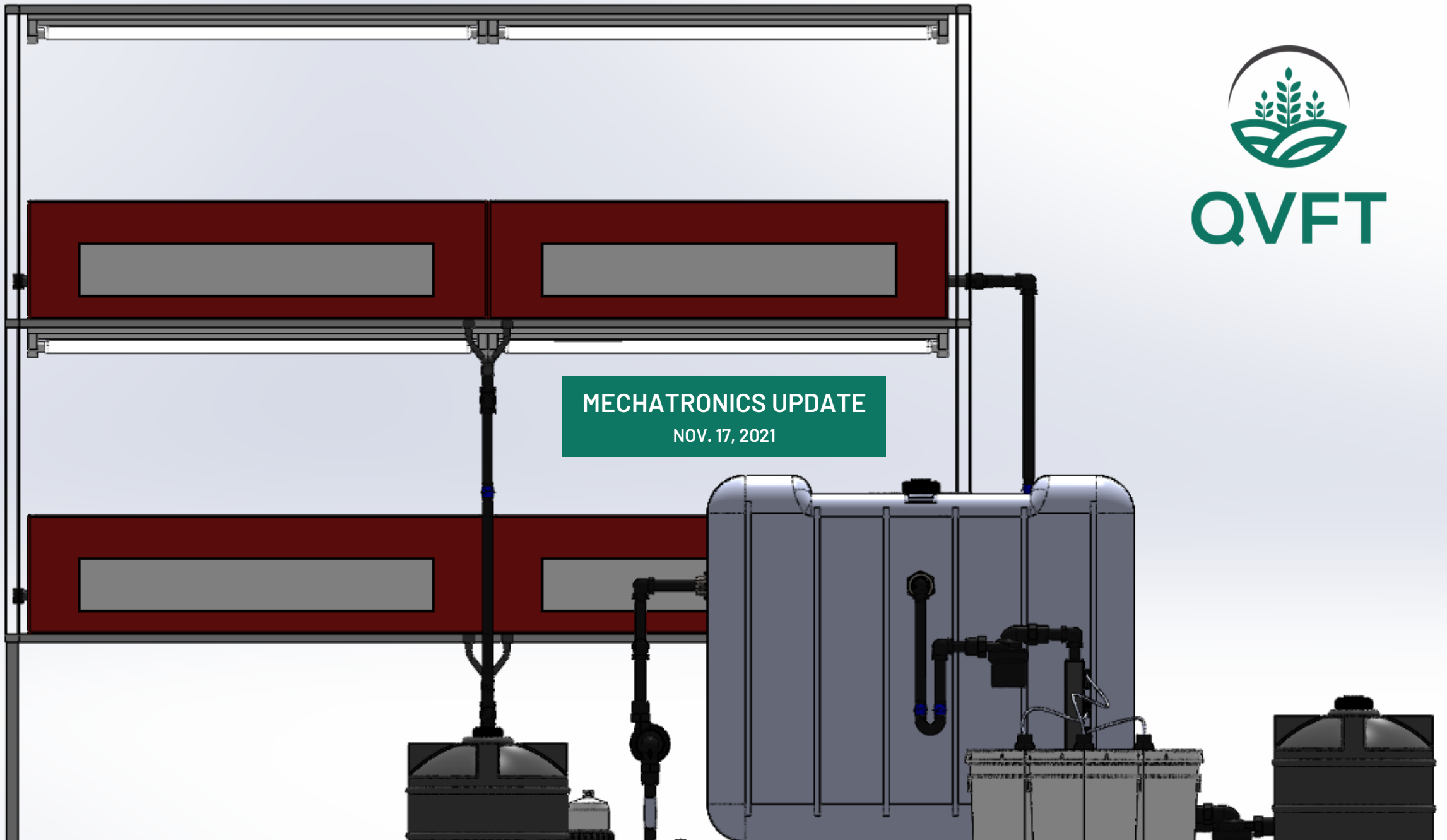
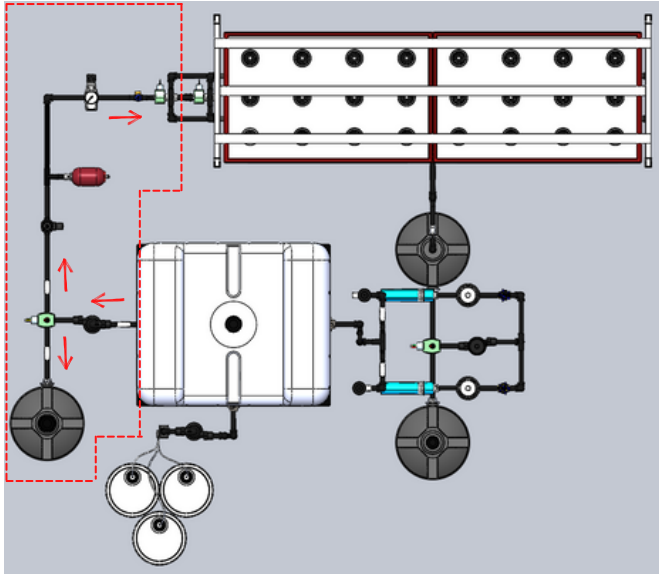
QVFT

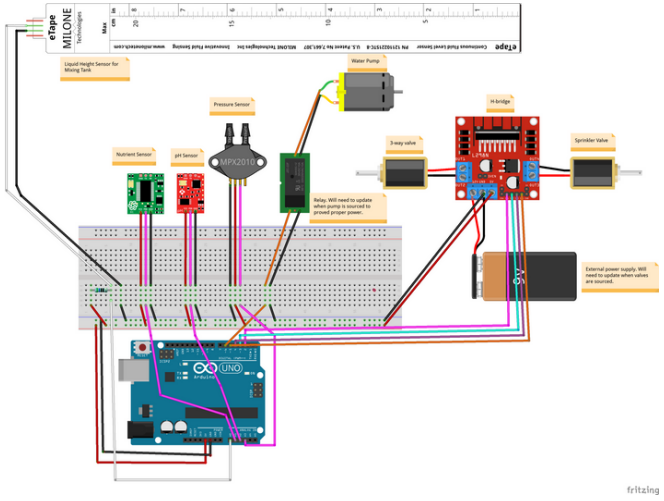
MECHATRONICS UPDATE

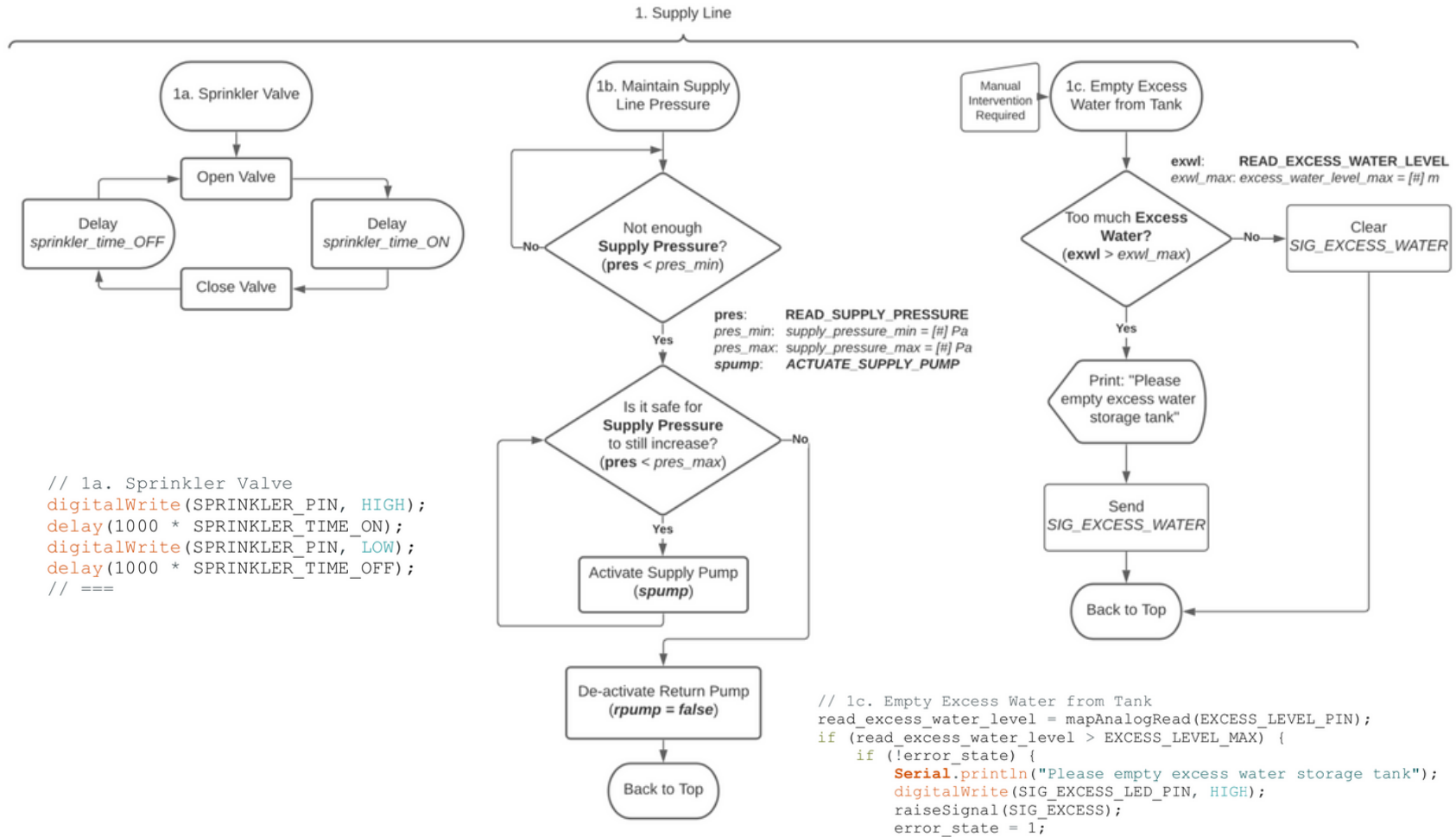
NOV. 17, 2021



SUPPLY LINE





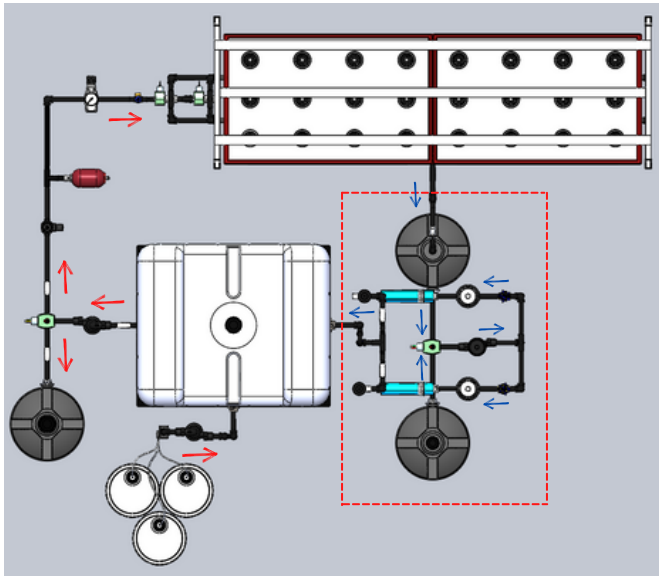


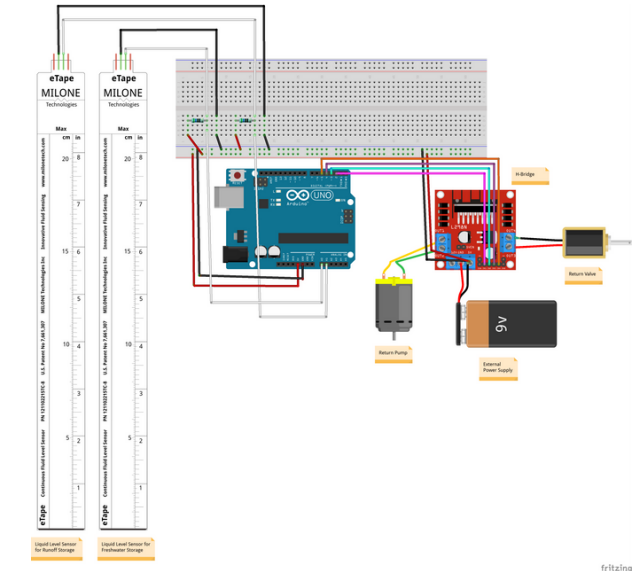

```
// 1a. Sprinkler Valve
digitalWrite(SPRINKLER_PIN, HIGH);
delay(1000 * SPRINKLER_TIME_ON);
digitalWrite(SPRINKLER_PIN, LOW);
delay(1000 * SPRINKLER_TIME_OFF);
// ===
```

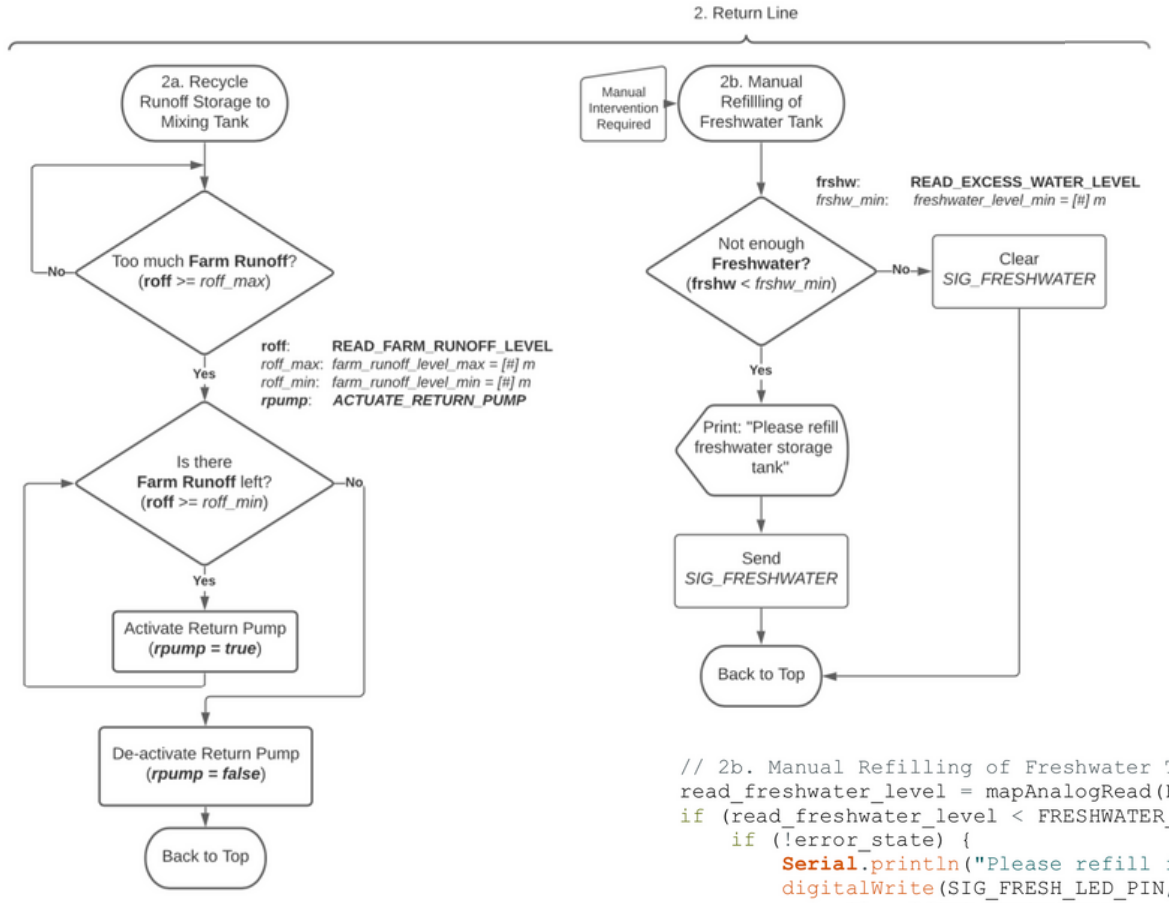
```
// 1b. Maintain Supply Line Pressure
read_supply_pressure = mapAnalogRead(SUPPLY_PUMP_PIN);
if (read_supply_pressure < SUPPLY_PRESSURE_MIN) {

    digitalWrite(SUPPLY_PUMP_PIN, HIGH);
    while (read_supply_pressure < SUPPLY_PRESSURE_MAX) {
        delay(500);
    }
    digitalWrite(SUPPLY_PUMP_PIN, LOW);

}
delay(1000 * WAIT_INTERVAL);
```

```
// 1c. Empty Excess Water from Tank
read_excess_water_level = mapAnalogRead(EXCESS_LEVEL_PIN);
if (read_excess_water_level > EXCESS_LEVEL_MAX) {
    if (!error_state) {
        Serial.println("Please empty excess water storage tank");
        digitalWrite(SIG_EXCESS_LED_PIN, HIGH);
        raiseSignal(SIG_EXCESS);
        error_state = 1;
    }
} else {
    if (error_state) {
        digitalWrite(SIG_EXCESS_LED_PIN, LOW);
        clearSignal(SIG_EXCESS);
        error_state = 0;
    }
}
delay(1000 * SHORT_WAIT_INTERVAL);
```

RETURN LINE



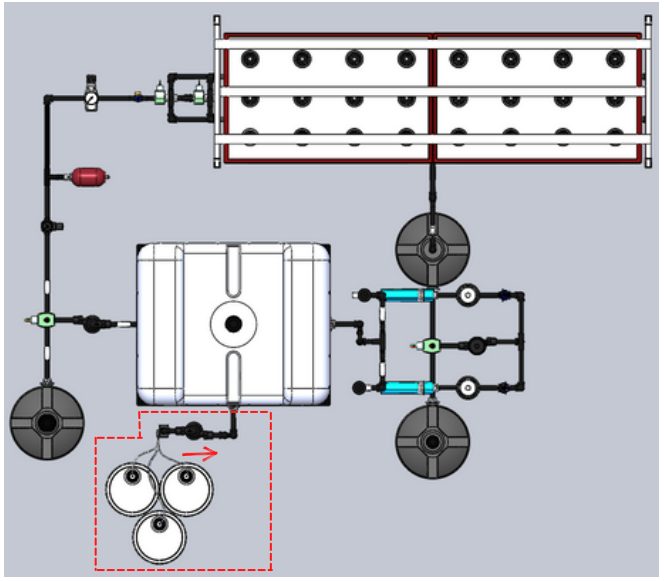


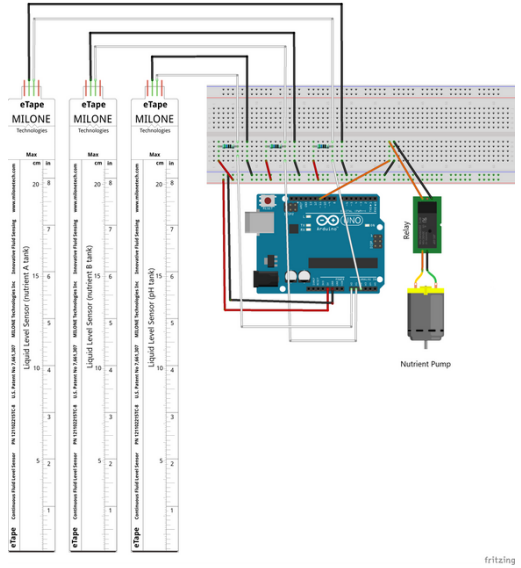


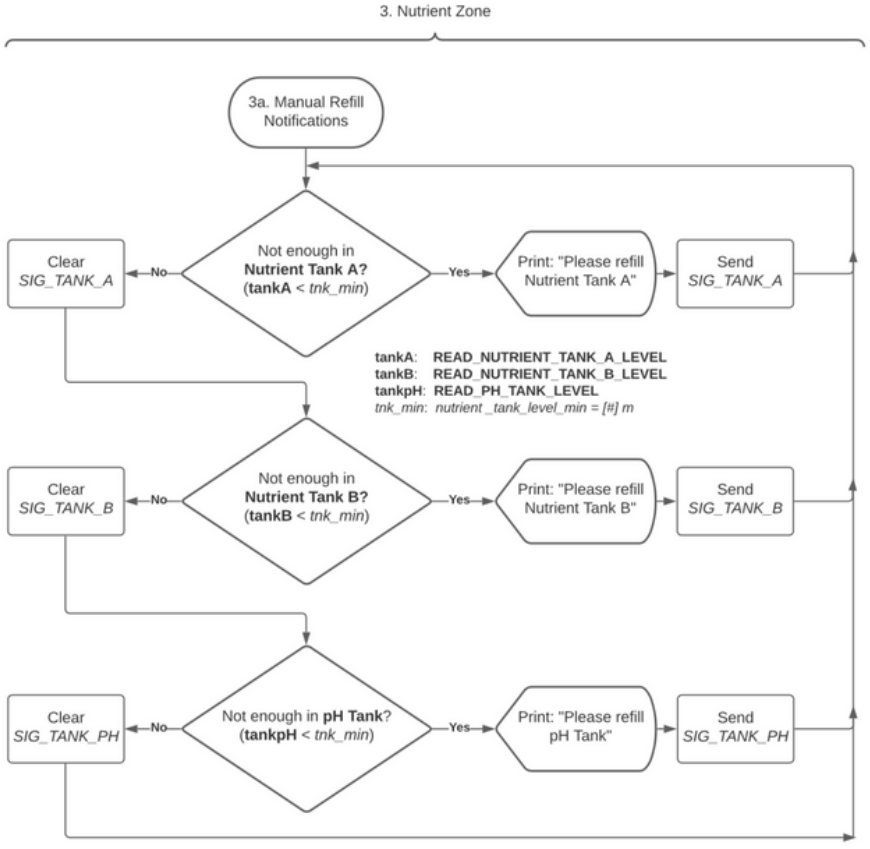

```
// 2a. Recyle Runoff Storage to Mixing Tank
read_farm_runoff_level = mapAnalogRead(RUNOFF_LEVEL_PIN);
if (read_farm_runoff_level >= RUNOFF_LEVEL_MAX) {

    digitalWrite(RETURN_PUMP_PIN, HIGH);

    while (read_farm_runoff_level >= RUNOFF_LEVEL_MIN) {
        delay(500);
    }
    digitalWrite(RETURN_PUMP_PIN, LOW);

}
delay(1000 * WAIT_INTERVAL);
```

```
// 2b. Manual Refilling of Freshwater Tank
read_freshwater_level = mapAnalogRead(FRESHWATER_LEVEL_PIN);
if (read_freshwater_level < FRESHWATER_LEVEL_MIN) {
    if (!error_state) {
        Serial.println("Please refill freshwater storage tank");
        digitalWrite(SIG_FRESH_LED_PIN, HIGH);
        raiseSignal(SIG_FRESHWATER);
        error_state = 1;
    }
} else {
    if (error_state) {
        digitalWrite(SIG_FRESH_LED_PIN, LOW);
        clearSignal(SIG_FRESHWATER);
        error_state = 0;
    }
}
delay(1000 * SHORT_WAIT_INTERVAL);
```

NUTRIENT ZONE

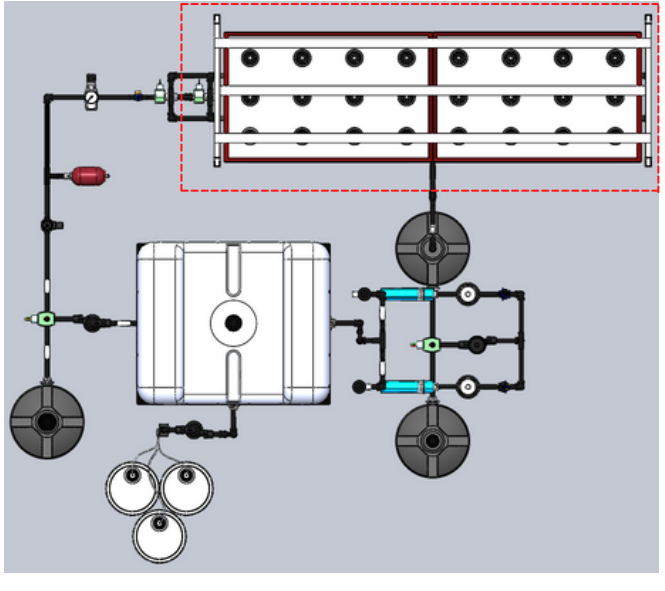


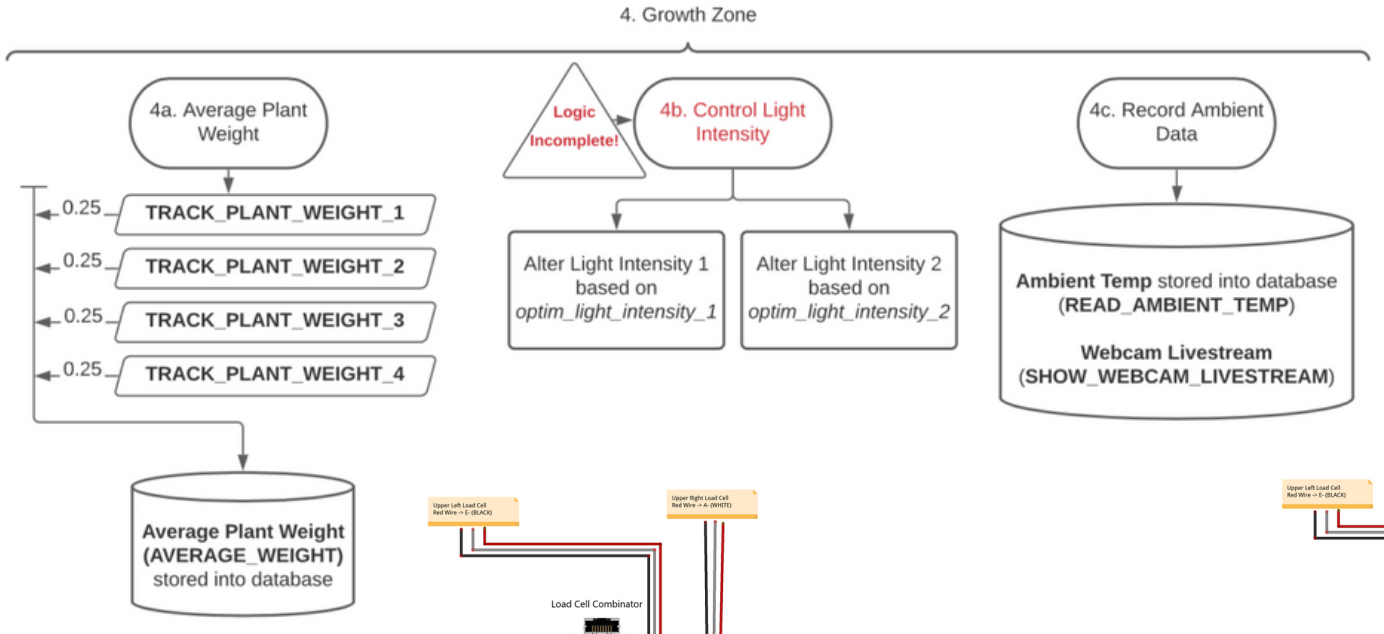


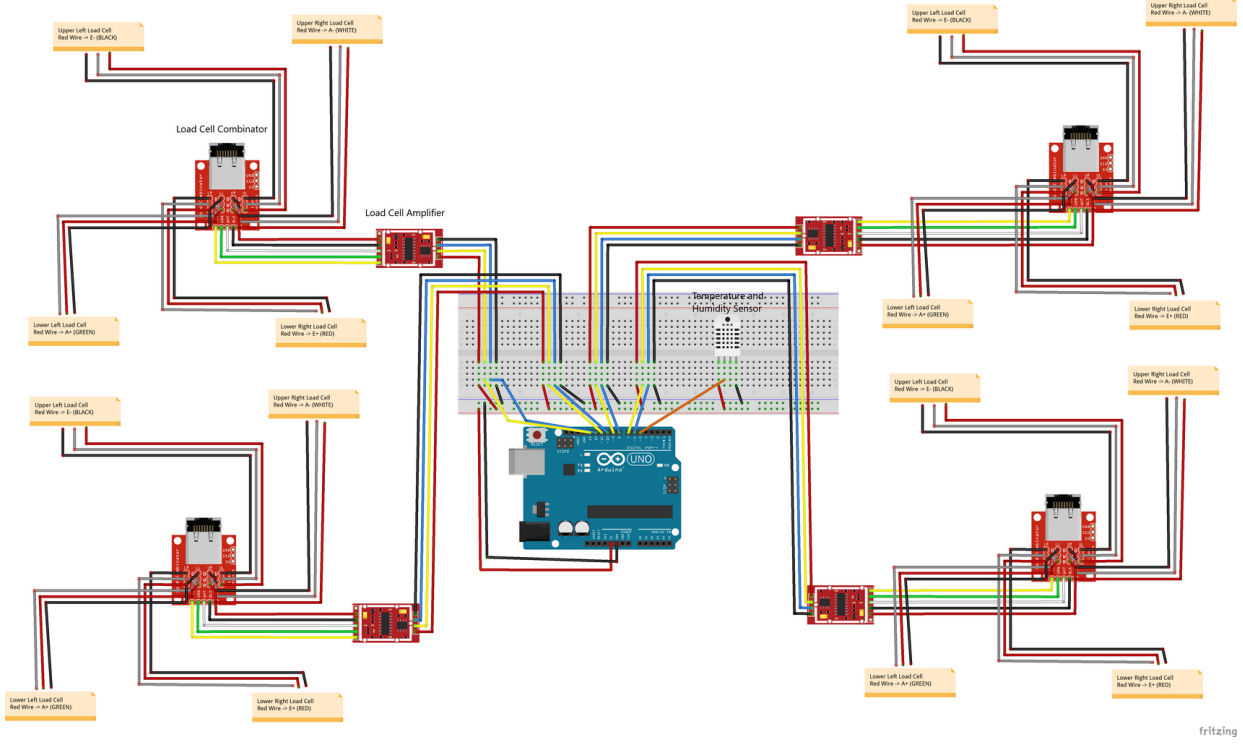



```
// 3a. Manual Refill Notifications
double read_nutrient_tankA_level = 0;
double read_nutrient_tankB_level = 0;
double read_tankPH_level = 0;
if (read_nutrient_tankA_level < NUTRIENT_TANK_LEVEL_MIN) {
    if (!error_state[0]) {
        Serial.println("Please refill Nutrient Tank A");
        raiseSignal(SIG_TANK_A);
        error_state[0] = 1;
    }
} else {
    if (error_state[0]) {
        clearSignal(SIG_TANK_A);
        error_state[0] = 0;
    }
}

if (read_nutrient_tankB_level < NUTRIENT_TANK_LEVEL_MIN) {
    if (!error_state[1]) {
        Serial.println("Please refill Nutrient Tank B");
        raiseSignal(SIG_TANK_B);
        error_state[1] = 1;
    }
} else {
    if (error_state[1]) {
        clearSignal(SIG_TANK_B);
        error_state[1] = 0;
    }
}

if (read_pH_tank_level < NUTRIENT_TANK_LEVEL_MIN) {
    if (!error_state[2]) {
        Serial.println("Please refill pH Tank");
        raiseSignal(SIG_TANK_PH);
        error_state[2] = 1;
    }
} else {
    if (error_state[2]) {
        clearSignal(SIG_TANK_PH);
        error_state[2] = 0;
    }
}

if ( (error_state[0] + error_state[1] + error_state[2]) > 0 ) {
    // there is an error state somewhere, we are only using one LED

    if (!digitalRead(SIG_TANK_LED_PIN)) {
        digitalWrite(SIG_TANK_LED_PIN, HIGH);
    }
} else {
    if (digitalRead(SIG_TANK_LED_PIN)) {
        digitalWrite(SIG_TANK_LED_PIN, LOW);
    }
}

}
```

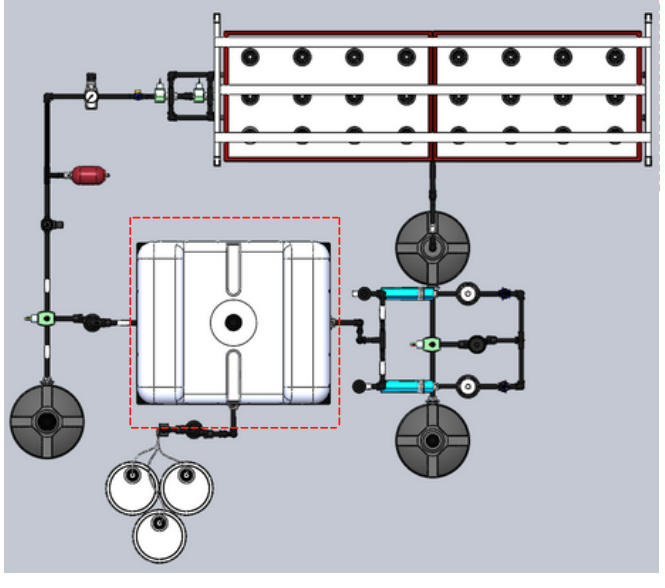
GROWTH ZONE

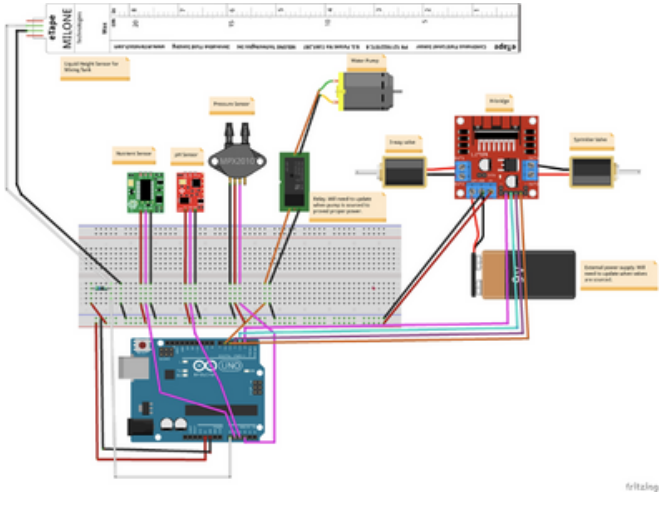


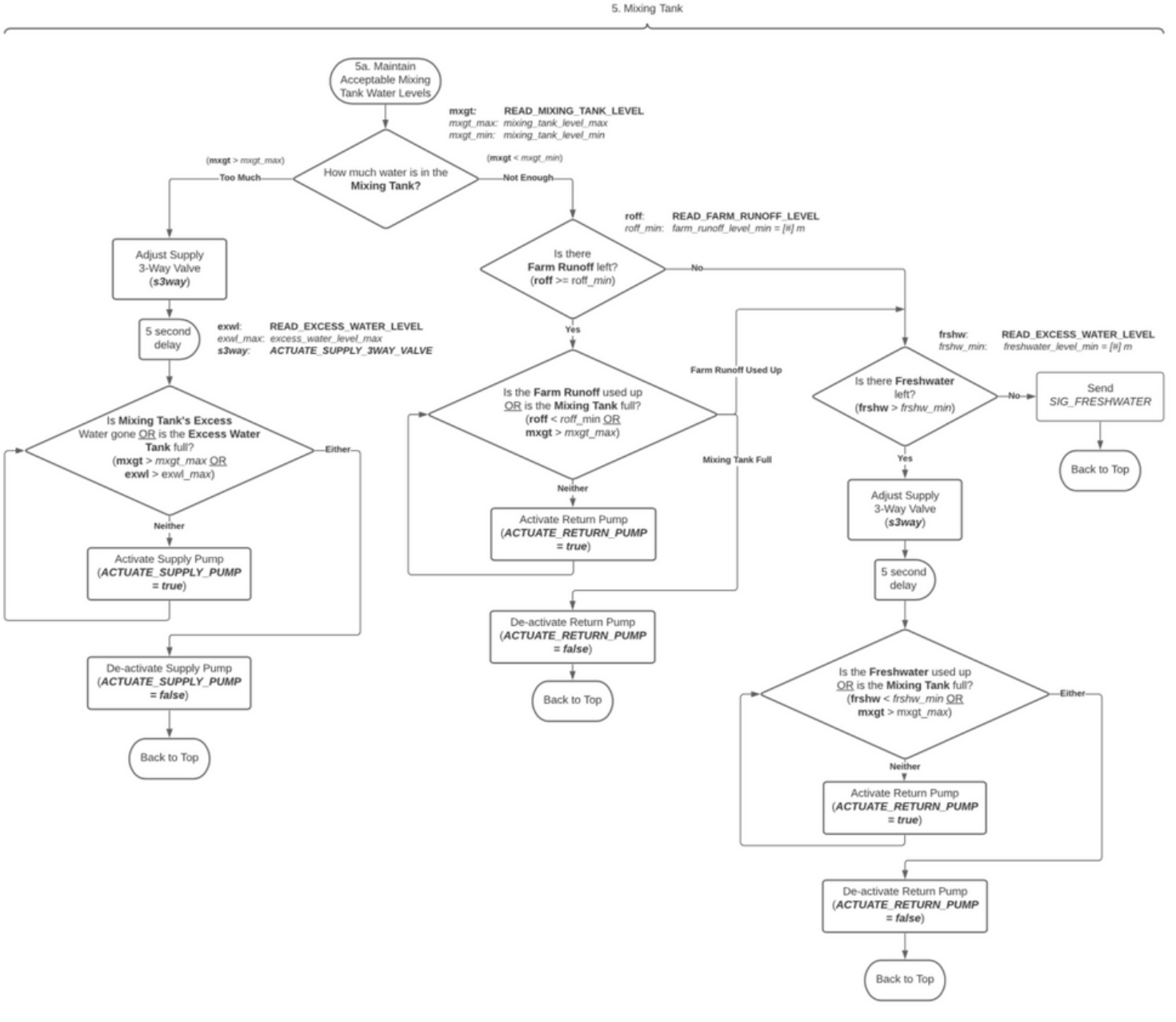

4. Growth Zone
4a. Average Plant Weight
0.25
TRACK_PLANT_WEIGHT_1
0.25
TRACK_PLANT_WEIGHT_2
0.25
TRACK_PLANT_WEIGHT_3
0.25
TRACK_PLANT_WEIGHT_4
Average Plant Weight (AVERAGE_WEIGHT) stored into database
Logic Incomplete!
4b. Control Light Intensity
Alter Light Intensity 1 based on optim_light_intensity_1
Alter Light Intensity 2 based on optim_light_intensity_2
4c. Record Ambient Data
Ambient Temp stored into database (READ_AMBIENT_TEMP)
Webcam Livestream (SHOW_WEBCAM_LIVESTREAM)


Load Cell Combinator
Load Cell Amplifier
Temperature and Humidity Sensor
fritzing

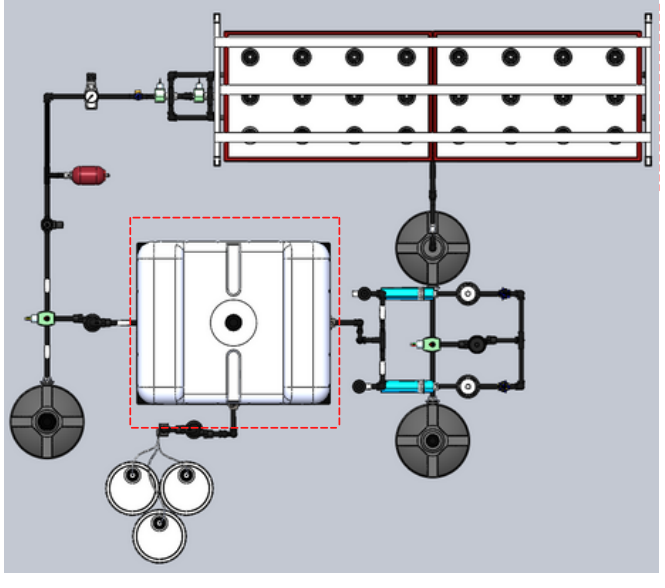
MIXING TANK (1)

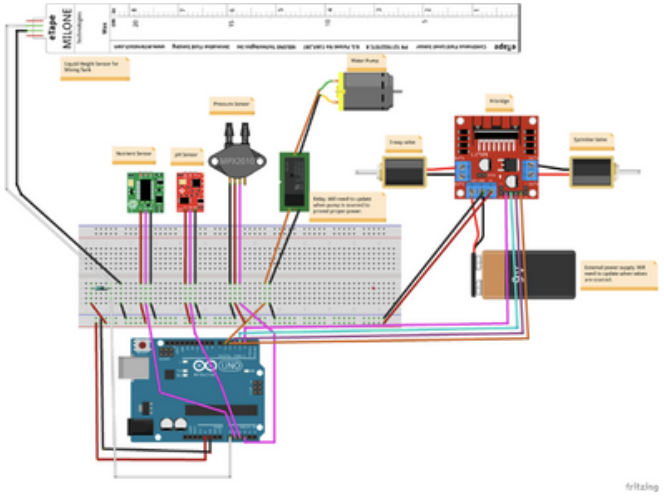


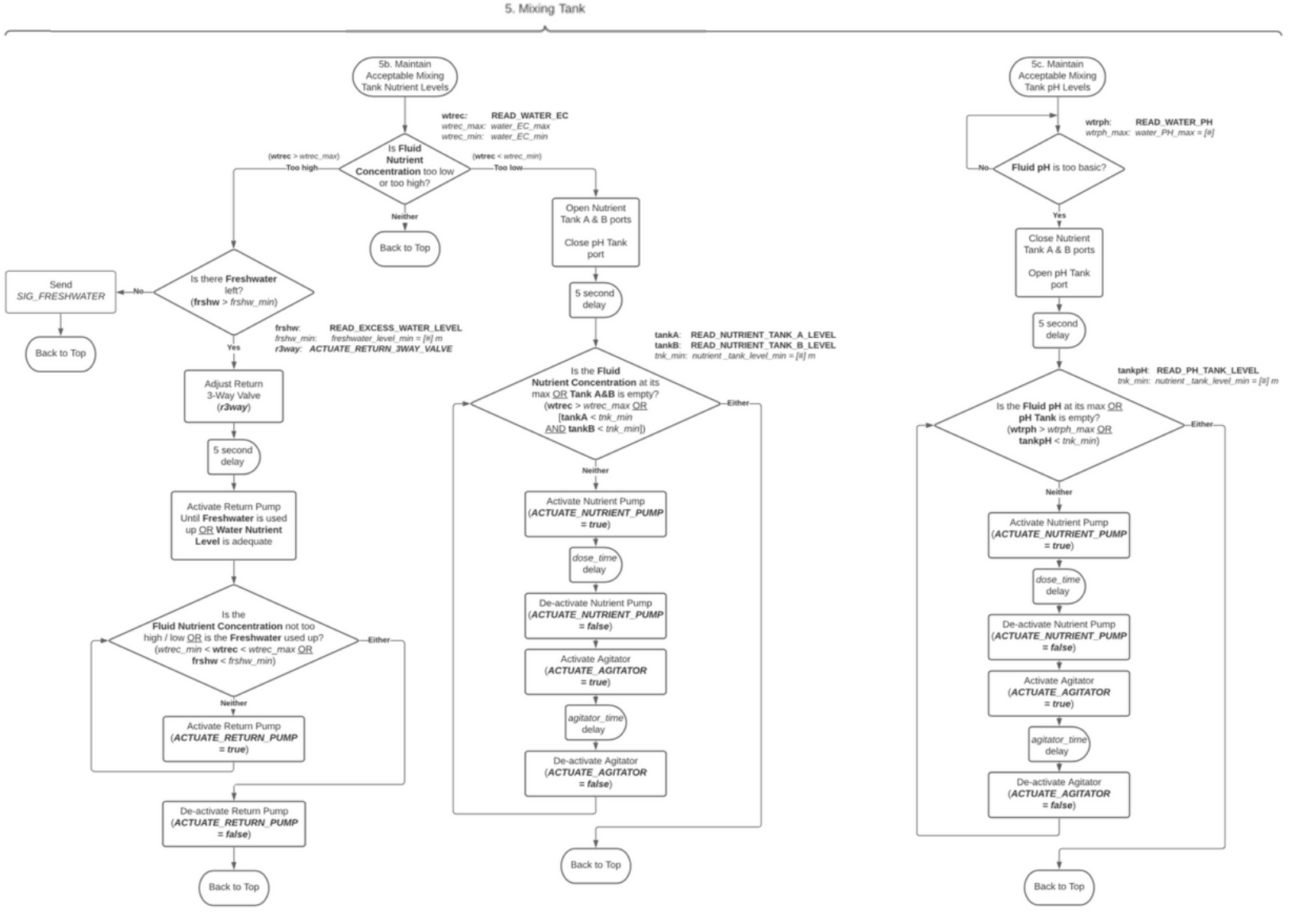




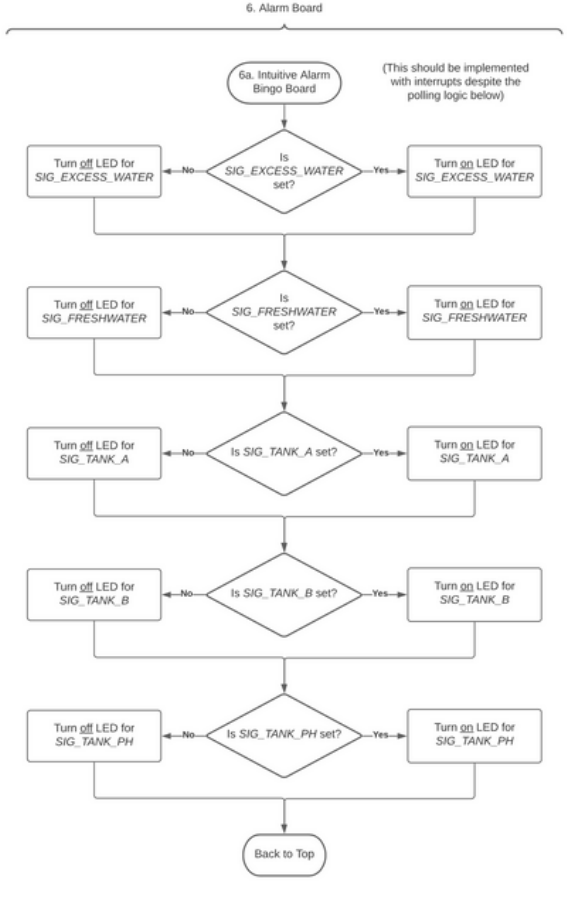
MIXING TANK (2)







ADDITIONAL INFORMATION




```
/*****************************************
  QVFT Farm Feedback-Control System
(Preliminary Code)
  Original by Q.Hu/R.Power/P.Singal, 11/2021
*****************************************/

#import "variables.h"

#define ERR_LED_PIN 13 // unset
#define WAIT_INTERVAL 60 // unset
#define SHORT_WAIT_INTERVAL 15 // unset

// 1a
#define SPRINKLER_PIN 6
#define SPRINKLER_TIME_ON 5 // seconds, unset
#define SPRINKLER_TIME_OFF 5 // seconds,
unset

// 1b
#define SUPPLY_PRESSURE_PIN A3
#define SUPPLY_PRESSURE_MIN 0 // Pa -> mV,
unset
#define SUPPLY_PRESSURE_MAX 1 // Pa -> mV,
unset
#define SUPPLY_PUMP_PIN 0 // unset
double read_supply_pressure = 0.0;

// 1c
#define EXCESS_LEVEL_PIN A0
#define EXCESS_LEVEL_MAX 0 // m -> mV, unset
#define SIG_EXCESS_LED_PIN 13 // unset
#define SIG_EXCESS 0
double read_excess_water_level = 0.0;
unsigned char error_state = 0;

// 2a
#define RUNOFF_LEVEL_PIN A1 // unset
#define RUNOFF_LEVEL_MIN 0 // m -> mV, unset
#define RUNOFF_LEVEL_MAX 0 // m -> mV, unset
#define RETURN_PUMP_PIN 0 // unset
double read_farm_runoff_level = 0.0;

// 2b
#define FRESHWATER_LEVEL_PIN A0 // unset
#define FRESHWATER_LEVEL_MIN 0 // m -> mV,
unset
#define SIG_FRESH_LED_PIN 13 // unset
#define SIG_FRESHWATER 1
double read_freshwater_level = 0.0;
unsigned char error_state = 0;

// 3a
#define NUTRIENT_TANK_LEVEL_MIN 0 // m -> mV,
unset
#define SIG_TANK_LED_PIN 13 // unset
#define SIG_TANK_A 2
#define SIG_TANK_B 3
#define SIG_TANK_PH 4
double read_nutrient_tankA_level = 0.0;
double read_nutrient_tankB_level = 0.0;
double read_pH_tank_level = 0.0;
unsigned char error_state[] = {0, 0, 0};

void raiseSignal(int signal) {
    // use wifi
}

void clearSignal(int signal) {
    // use wifi
}

double mapAnalogRead(int pin) {
    return map(analogRead(pin), 0, 1023, 0,
5);
}

void setup() {
    Serial.begin(9600);

    // WiFi connection here

    // 1a.
    pinMode(SPRINKLER_PIN, OUTPUT);
    digitalWrite(SPRINKLER_PIN, LOW);
    // ===

    // 1b.
    pinMode(SUPPLY_PUMP_PIN, OUTPUT);
    digitalWrite(SUPPLY_PUMP_PIN, LOW);

    // 1c.
    pinMode(SIG_EXCESS_LED_PIN, OUTPUT);
    digitalWrite(SIG_EXCESS_LED_PIN, LOW);

    // 2c.
    pinMode(RETURN_PUMP_PIN, OUTPUT);
    digitalWrite(RETURN_PUMP_PIN, HIGH);

    // 3a.
    pinMode(SIG_TANK_LED_PIN, OUTPUT);
    digitalWrite(SIG_TANK_LED_PIN, LOW);

}

void loop() {

    // 1a. Sprinkler Valve
    digitalWrite(SPRINKLER_PIN, HIGH);
    delay(1000 * SPRINKLER_TIME_ON);
    digitalWrite(SPRINKLER_PIN, LOW);
    delay(1000 * SPRINKLER_TIME_OFF);
    // ===

    // 1b. Maintain Supply Line Pressure
    read_supply_pressure =
mapAnalogRead(SUPPLY_PUMP_PIN);
    if (read_supply_pressure <
SUPPLY_PRESSURE_MIN) {

        digitalWrite(SUPPLY_PUMP_PIN, HIGH);
        while (read_supply_pressure <
SUPPLY_PRESSURE_MAX) {
            delay(500);
        }
        digitalWrite(SUPPLY_PUMP_PIN, LOW);

    }
    delay(1000 * WAIT_INTERVAL);

    // 1c. Empty Excess Water from Tank
    read_excess_water_level =
mapAnalogRead(EXCESS_LEVEL_PIN);
    if (read_excess_water_level >
EXCESS_LEVEL_MAX) {
        if (!error_state) {
            Serial.println("Please empty
excess water storage tank");
            digitalWrite(SIG_EXCESS_LED_PIN,
HIGH);
            raiseSignal(SIG_EXCESS);
            error_state = 1;
        }
    } else {
        if (error_state) {
            digitalWrite(SIG_EXCESS_LED_PIN,
LOW);
            clearSignal(SIG_EXCESS);
            error_state = 0;
        }
    }
    delay(1000 * SHORT_WAIT_INTERVAL);

    // 2a. Recyle Runoff Storage to Mixing
Tank
    read_farm_runoff_level =
mapAnalogRead(RUNOFF_LEVEL_PIN);
    if (read_farm_runoff_level >=
RUNOFF_LEVEL_MAX) {

        digitalWrite(RETURN_PUMP_PIN, HIGH);
        while (read_farm_runoff_level >=
RUNOFF_LEVEL_MIN) {
            delay(500);
        }
        digitalWrite(RETURN_PUMP_PIN, LOW);

    }
    delay(1000 * WAIT_INTERVAL);

    // 2b. Manual Refilling of Freshwater
Tank
    read_freshwater_level =
mapAnalogRead(FRESHWATER_LEVEL_PIN);
    if (read_freshwater_level <
FRESHWATER_LEVEL_MIN) {
        if (!error_state) {
            Serial.println("Please refill
freshwater storage tank");
            digitalWrite(SIG_FRESH_LED_PIN,
HIGH);
            raiseSignal(SIG_FRESHWATER);
            error_state = 1;
        }
    } else {
        if (error_state) {
            digitalWrite(SIG_FRESH_LED_PIN,
LOW);
            clearSignal(SIG_FRESHWATER);
            error_state = 0;
        }
    }
    delay(1000 * SHORT_WAIT_INTERVAL);

    // 3a. Manual Refill Notifications
    double read_nutrient_tankA_level = 0;
    double read_nutrient_tankB_level = 0;
    double read_tankPH_level = 0;
    if (read_nutrient_tankA_level <
NUTRIENT_TANK_LEVEL_MIN) {
        if (!error_state[0]) {
            Serial.println("Please refill
Nutrient Tank A");
            raiseSignal(SIG_TANK_A);
            error_state[0] = 1;
        }
    } else {
        if (error_state[0]) {
            clearSignal(SIG_TANK_A);
            error_state[0] = 0;
        }
    }

    if (read_nutrient_tankB_level <
NUTRIENT_TANK_LEVEL_MIN) {
        if (!error_state[1]) {
            Serial.println("Please refill
Nutrient Tank B");
            raiseSignal(SIG_TANK_B);
            error_state[1] = 1;
        }
    } else {
        if (error_state[1]) {
            clearSignal(SIG_TANK_B);
            error_state[1] = 0;
        }
    }

    if (read_pH_tank_level <
NUTRIENT_TANK_LEVEL_MIN) {
        if (!error_state[2]) {
            Serial.println("Please refill pH
Tank");
            raiseSignal(SIG_TANK_PH);
            error_state[2] = 1;
        }
    } else {
        if (error_state[2]) {
            clearSignal(SIG_TANK_PH);
            error_state[2] = 0;
        }
    }

    if ( (error_state[0] + error_state[1] +
error_state[2]) > 0 ) {
        // there is an error state somewhere,
we are only using one LED
        if (!digitalRead(SIG_TANK_LED_PIN)) {
            digitalWrite(SIG_TANK_LED_PIN,
HIGH);
        }
    } else {
        if (digitalRead(SIG_TANK_LED_PIN)) {
            digitalWrite(SIG_TANK_LED_PIN,
LOW);
        }
    }
}
```