CENTER FOR DEVELOPMENT OF ELECTRONIC DEVICES

Promoting a culture of innovation

THE SMART WATER METER PROJECT

DATE: 29TH OCT 2023

THE FINAL SMART WATER METER PROJECT PROTOTYPE PROGRAM

The program is developed in the ArduinoIDE using the new STM32Duino programming procedure. The STM32 Microcontroller is programmed using the Arduino IDE via the STM32 Core Library developed by STM32Duino. This library enables STM32 microcontrollers to be programmed using the Arduino IDE and upload the program via an ST-Link.

The program has been tested and confirmed to work as described and commented.

This program interfaces STM32F103C8T6 with the following components:

- Water Flow Sensor
- Motorized Ball Valve via L298N Driver
- HT1621 LCD
- IR Sensor
- GSM SIM7600CE 4G Module

WORKING

The program continuously records pulses generated by the water flow sensor through the Interrupt Service Routine and utilizes the equation $Volume = (0.002663 * pulses)$ to calculate the corresponding volume in liters of water passing through the sensor. The value 0.002663 was measured practically to determine what volume of water corresponded to one single pulse of the water flow sensor. The Volume is then continuously displayed on the HT1621 LCD and incremented in real time as water is being consumed. The Volume of water consumed in 24 hours is sent as an SMS to the phone number specified in the program (+254796456877), for record-keeping. In the event the water supplier wants to remotely cut off water supply, an SMS is sent to the Meter, and it is executed to rotate the motorized ball valve in the clockwise direction, thereby cutting the supply of water. To reduce power consumption, the IR sensor controls the switching on and off of the HT1621. This is affected by a cover lid on the meter. When the cover lid is lifted, the LCD will display data, but when it is closed, it will be shut off and run in the background.

TO BE ADDED:

EEPROM to store data in the event power is suddenly lost before an SMS is sent. The memory could not be added now because STM32Duino has not yet developed an I2C protocol for STM32 in Arduino. Once the library is officially released, EEPROM can be included to increase the efficiency of the Meter.

BATTERY POWER DRAINAGE

The Battery Markers in the HT1621 LCD can be customized as follows to monitor how the battery power is being drained as the meter is running:

Battery Capacity = 4000 mAh (an approximate of the actual battery)

Total current drawn while the meter is running (Motor not moving and no SMS being sent):

= 15mAh (An approximate, the actual current drawn can be measured directly from the battery as the meter is running).

Actual Battery capacity will be 4000mAh less the total power consumed by the meter when sending an SMS and power consumed by the motor when turning on or off the meter. Since these two are not continuous in the normal meter running, their effect can be accounted for after the calculations.

Time it will take the battery to go down:

$$= \frac{4000\ mAh}{15\ mAh} = 267\ hours\ or\ 11\ Days$$

This is so by assuming the meter is running continuously 24/7 which is not the case. Since during the night the meter will be in idle mode and during the day, not all the time the water flow sensor will be running, this number can be increased by approximately 1/2. The new duration will be 22

Days. We can settle with this number in order to work with the minimum value. This will provide a grace period to change the battery before the meter goes off.

To account for the SMS power consumption and Motor power consumption if it is turned on or off, an approximate of 20 days will be somehow accurate.

Setting the delay time for the Battery Markers:

$$\frac{(480\ hours)}{3} = 160\ hours = 576000000\ milliseconds$$

Therefore, each marker should last approximately 160 hours after the former fades out. This program is set for this value. After the last marker fades out, no marker will be displayed.

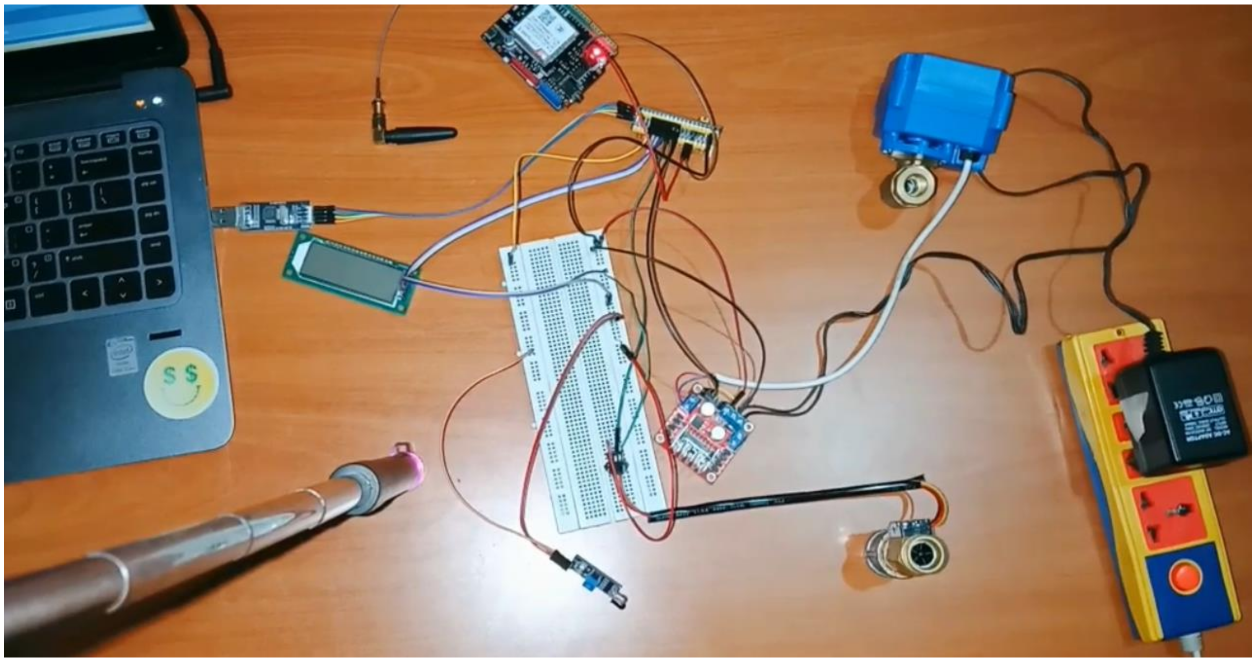
The approximated values can be replaced with measured values to achieve higher accuracy.

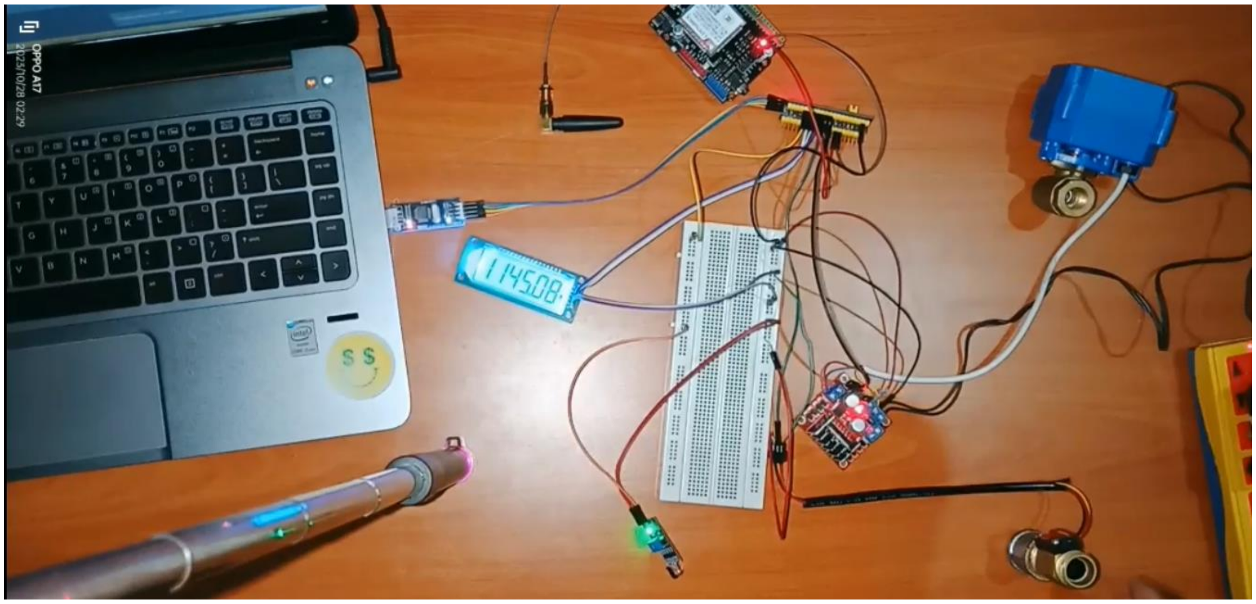
Program developed by Eric Mulwa, BSc EEE.

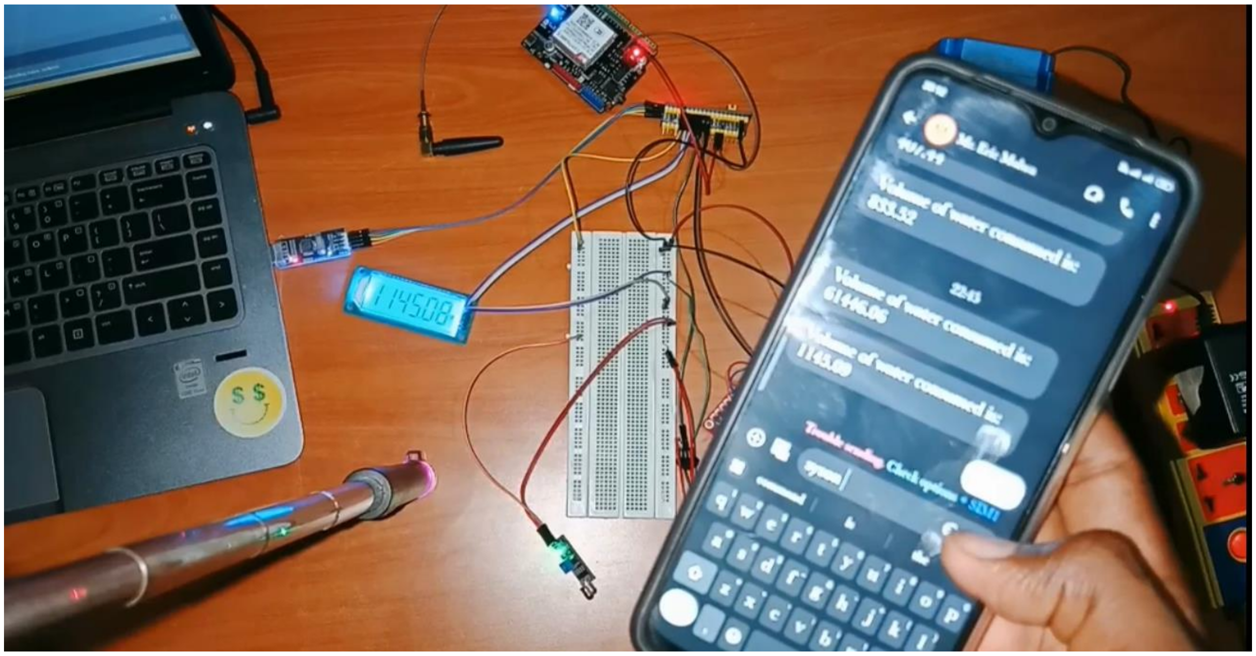
```
#include <HT1621.h> // Include our library
HT1621 lcd; // Create an "lcd" object
#define IrPin 0 // Define the IR sensor pin as PA0
#define DrivercPin 5 // PA5 for opening the Ball-valve
#define DriveraPin 6 // PA6 for closing the Ball-valve

int sensorPin = 4; // PA4
volatile long pulse;
float volume;
unsigned long lastSMSTime = 0; // Timer for the last SMS sent
const unsigned long smsInterval = 86400000; // 24 hours in milliseconds

char incomingMessage[100]; // Array to store incoming SMS
unsigned long batteryMarkerTimer = 0;
int batteryMarkerState = 3; // Initial state
bool batteryMarkersExecuted = false; // Flag to control battery markers execution

void setup() {
  pinMode(sensorPin, INPUT);
  pinMode(IrPin, INPUT); // Set the IR sensor pin as input
  pinMode(DrivercPin, OUTPUT);
  pinMode(DriveraPin, OUTPUT);
  attachInterrupt(digitalPinToInterrupt(sensorPin), increase, RISING);

  Serial1.begin(115200); // Initialize GSM serial communication
  delay(1000); // Wait for the GSM module to start
  // Initialize the GSM module
  Serial1.println("AT");
  delay(1000);
  Serial1.println("AT+CMGF=1"); // Set SMS mode to text
  delay(1000);
  Serial1.println("AT+CNMI=2,2,0,0,0"); // Enable SMS notifications
  delay(1000);

  // Start the LCD:
  // CS to pin PA1
  // WR to pin PA2
  // Data to pin PA3
  lcd.begin(PA1, PA2, PA3); // (CS, WR, Data)
  lcd.clear();

  // Set the last SMS time to the current time to ensure the first SMS is sent
after 24 hours
  lastSMSTime = millis();
}
```

```
void loop() {
  // Check for incoming SMS
  if (Serial1.available()) {
    char c = Serial1.read();
    if (c == '\n') {
      // Check the received message
      if (strstr(incomingMessage, "syson") != NULL) {
        digitalWrite(DrivercPin, HIGH); // Open the motorized ball-valve
        delay(3000); //
        digitalWrite(DrivercPin, LOW);
      } else if (strstr(incomingMessage, "sysoff") != NULL) {
        digitalWrite(DriveraPin, HIGH); // Close the motorized ball-valve
        delay(3000); // Wait for 2s
        digitalWrite(DriveraPin, LOW); //
      }
      // Clear the message buffer
      memset(incomingMessage, 0, sizeof(incomingMessage));
    } else {
      // Append the received character to the message buffer
      strncat(incomingMessage, &c, 1);
    }
  }

  // Calculate and display the volume value continuously
  //volume = 2.663 * pulse; // Volume in Milliliters // for testing purposes
  volume = 0.002663 * pulse; // Volume in Liters

  int detect = digitalRead(IrPin); // Read IR sensor status and store
  if (detect == LOW) {
    lcd.clear(); // Clear the screen if an obstacle is detected
    delay(1000);
  } else if (detect == HIGH) {
    lcd.print(volume, 1); // Display the volume with 1 decimal place
  }

  unsigned long currentMillis = millis();
  // Setting the Battery Markers without interfering with any other code section
  if (!batteryMarkersExecuted) {
    // Display the 3rd Markers for 160 hours. That is the 3rd marker to last for
160 hours
    if (currentMillis - batteryMarkerTimer <= 576000000) {
      lcd.setBatteryLevel(3);
    }
    // Display the 2nd Marker for 320 hours. That is 2nd marker to last 160 hours
after the 3rd fades out
    else if (currentMillis - batteryMarkerTimer <= 1152000000) {
      lcd.setBatteryLevel(2);
    }
    // Display the 1st Marker for 480 hours. That is 1st marker to last 160 hours
after the 2nd fades out
    else if (currentMillis - batteryMarkerTimer <= 1728000000) {
      lcd.setBatteryLevel(1);
    }
    // Display no marker after the 480 hours. Battery shutting down
    else if (currentMillis - batteryMarkerTimer <= 1728002000) {
    lcd.setBatteryLevel(0);
    }
    // Stop the execution
    else {
      batteryMarkersExecuted = true;
    }
  }
  // Check if it's time to send the volume SMS
  if (currentMillis - lastSMSTime >= smsInterval) {
    // Send the SMS
    sendVolumeSMS();
    // Update the last SMS time
    lastSMSTime = currentMillis;
  }
}

void increase() {
  pulse++;
}

void sendVolumeSMS() {
  char mobileNumber[] = "+254796456877"; // Mobile Number to send Volume of water
consumed.
  char ATcommand[80];
  uint8_t buffer[30] = {0};
  uint8_t ATisOK = 0;
  while (!ATisOK) {
    sprintf(ATcommand, "AT\r\n");
    Serial1.print(ATcommand);
    delay(1000);
    if (Serial1.find("OK")) {
      ATisOK = 1;
    }
    delay(1000);
  }
  sprintf(ATcommand, "AT+CMGF=1\r\n");
  Serial1.print(ATcommand);
  delay(100);
  Serial1.readBytes(buffer, sizeof(buffer));
  delay(1000);
  memset(buffer, 0, sizeof(buffer));
  sprintf(ATcommand, "AT+CMGS=\"%s\"\r\n", mobileNumber);
  Serial1.print(ATcommand);
  delay(100);
  Serial1.print("Volume of water consumed in Liters: ");
  Serial1.print(volume, 2);
  Serial1.write(0x1a); // Send the Ctrl+Z character to indicate the end of the
message
  delay(4000);
}

 //--------------------@ ERIC MULWA BSc EEE -------------------------//
```

PROTOTYPE VISUALS







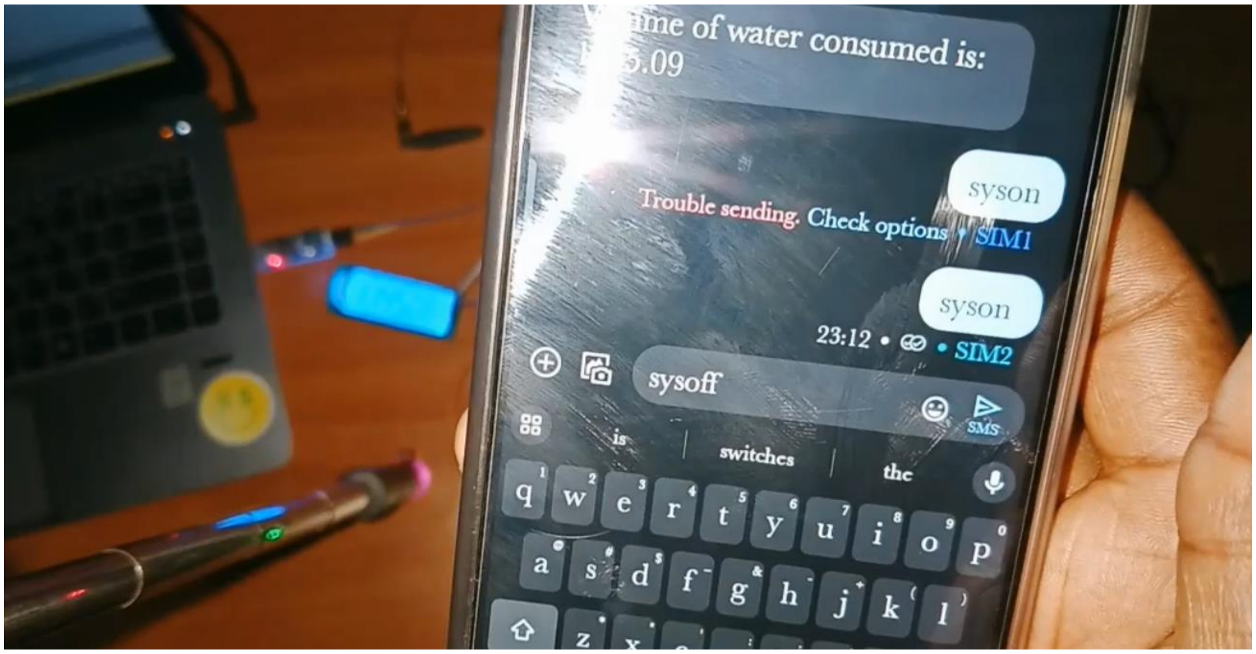
Volume of water consumed is:
833.52

22:43

Volume of water consumed is:
61446.06

Volume of water consumed is:
1143.09

Trouble sending. Check options • SIM1



me of water consumed is:
5.09

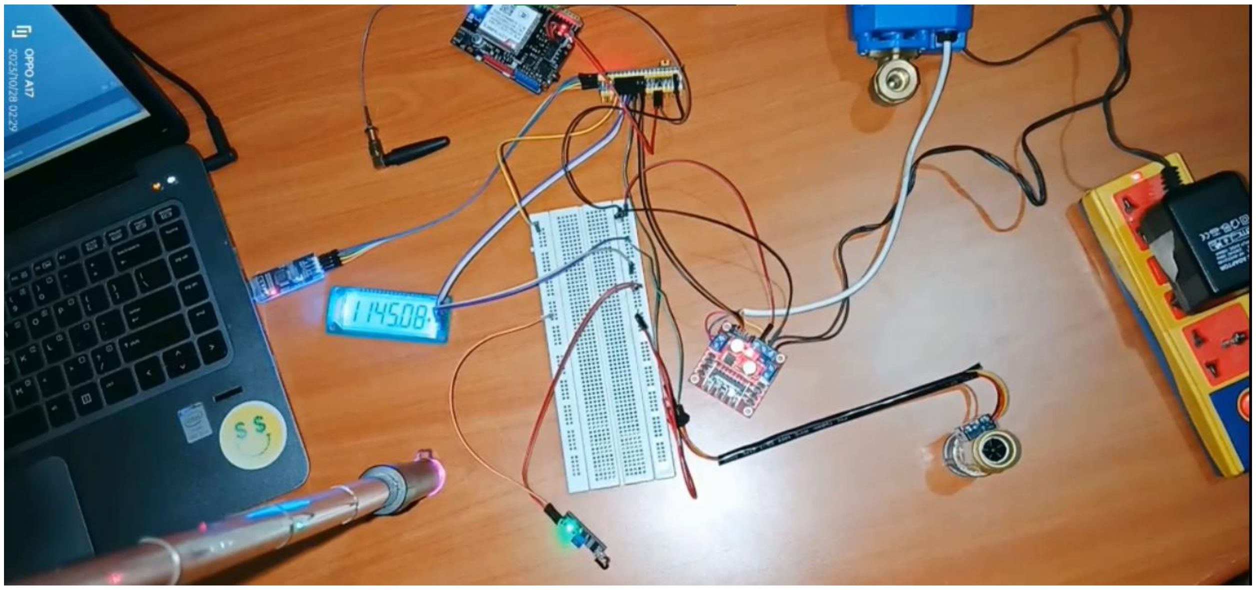
syson

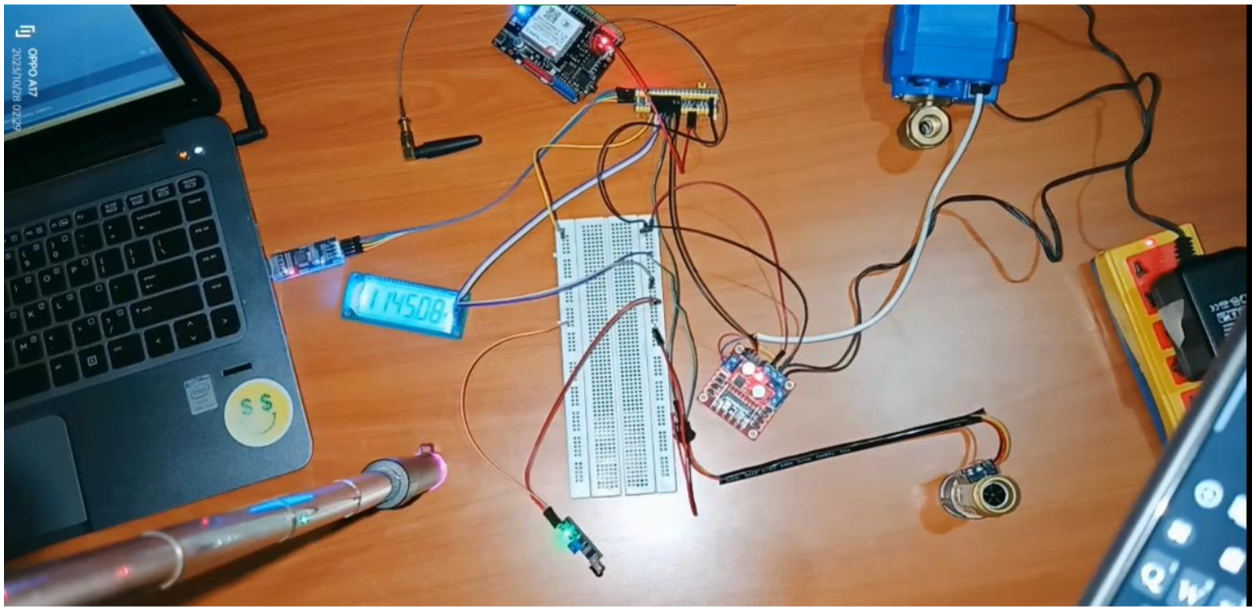
Trouble sending. Check options • SIM1

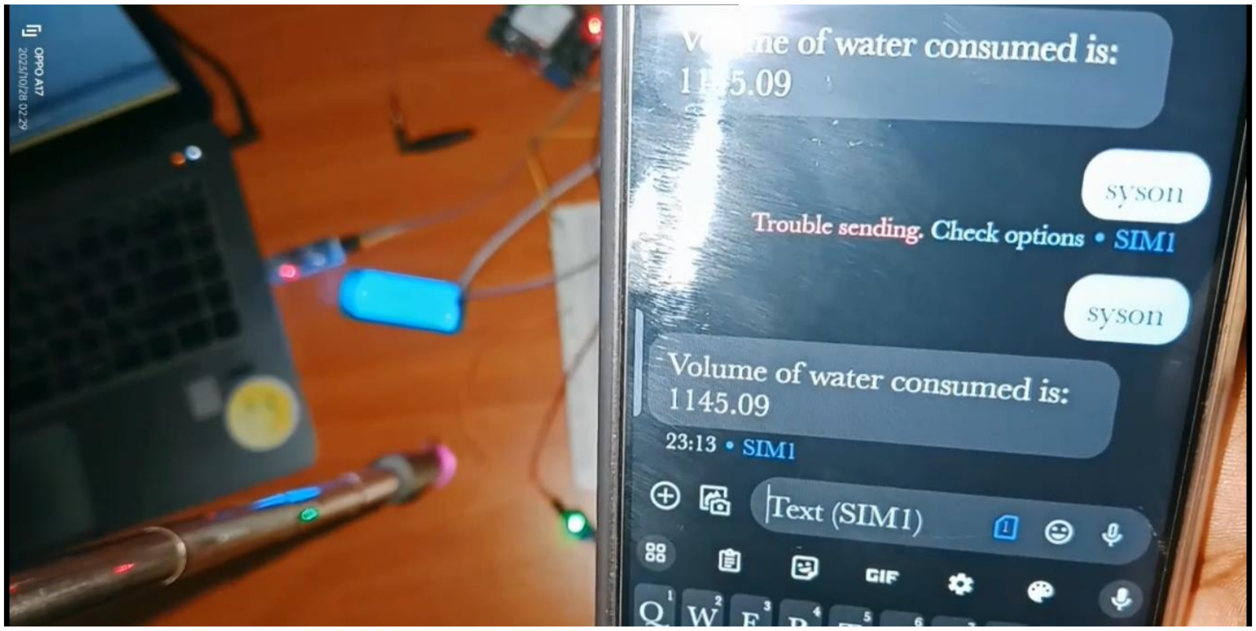
syson

23:12 • SIM2

sysoff


OPPO A17
2025/10/28 02:29
11450.8


OPPO A17
2025/10/28 02:29
11450.8



Volume of water consumed is:
1145.09

syson

Trouble sending. Check options • SIM1

syson

Volume of water consumed is:
1145.09

23:13 • SIM1