EEE3000X Report: Safi Manzi Water

Justin Cross: CRSKAT005

06/02/2024

Objectives:

Safi Manzi Water is a water filtration company which supplies products that deliver filtered water directly from your borehole or municipal source into your home. These filter units require backwashing processes to maintain the quality of the filters and the water that enters your home. I was tasked with finding and programming PLC solutions that would automate the filtration and backwashing processes to reduce manual maintenance procedures that would need to be done by the owner of the Safi unit.

Description of the equipment:

The Safi unit is comprised of four water filters, each that filter the water source before it enters your home. The filters are backwashed by closing a two-way ball valve which delivers the final filtered water into your home and having the valve to each filter consecutively be closed to create a cleaning mechanism that sends the filtered water from the other three filters through the closed filter and flush out any unwanted substances. The valves are controlled by electrical actuators that either close the flow of the water or switch the direction of the water flow to allow backwashed water to flow out of the unit into a drain.

Figure 1: Picture of automated Safi unit system.

The PLC system that was previously used was discontinued before the COVID era, hence new PLC solutions were required. Upon beginning the project, there were already two Raspberry Pi 4 units and displays that had been purchased that needed to be assembled and programmed for use with the Safi units. Other products tested was an Arduino Uno. I was also tasked with researching cost-effective PLC solutions and this is why the Arduino board was tested. It was also found that the Siemans Logic Logo was also an affordable option for the company.

Implementation:

Raspberry Pi 4:

The Raspberry Pi first needed to be assembled to create a working display along with the Pi board. I then used a breadboard along with 1K ohm resistors and LEDs to create a system that could simulate the backwashing process, while I created a python program to be used by the Pi on the Safi unit. I programmed the GPIO pins on the Pi board to turn the LEDs on and off in a sequence that replicated the desired actuator sequence and used a built-in python date and time function to customise the time that the Safi owner would want the system to be backwashed. The simulation ran successfully.

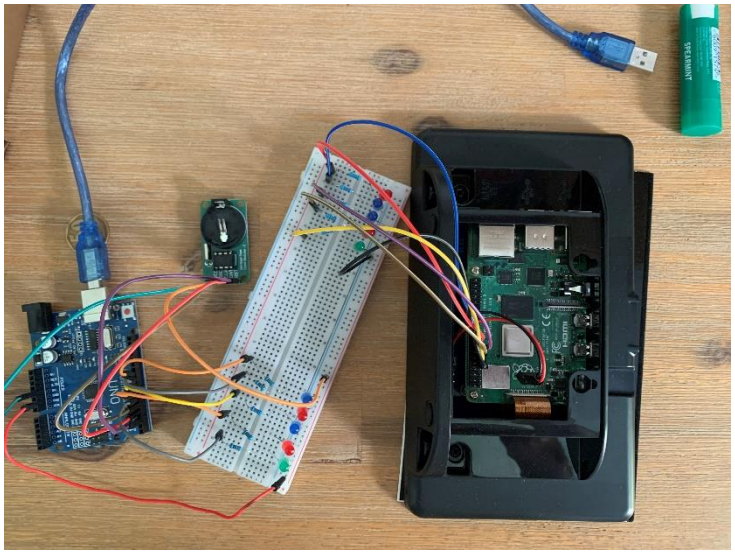
(See appendix A for python program)

Arduino Uno:

The same process was tested by writing an Arduino program that was uploaded to the Arduino Uno to replicate the same actuator switching process once again for the Safi. The Arduino did not have a built-in date and time function and so a RtcDS1302 was used to import the date and time to customise the timing schedule for the user. The program once again ran successfully.

(See appendix B for Arduino Program)

Figure 2: Raspberry Pi and Arduino simulation set-up.



Results:

The results obtained from the above programs were as expected and the programs ran with no bugs or timing issues. They allowed for customizable dates and times that the buyer of Safi unit could decide upon for the backwashing of their unit. The electrical equipment required for the assembly of the complete PLC system is still being delivered and the complete system will be tested upon delivery. I have also taken a course to understand the Siemans programming platform as the director of the project intends on using the Siemans Logic Logo in the future.

Other tasks:

Aside from the PLC development project, I was also involved with on-site fault finding and repairs of already purchased and assembled automated Safi units. This involved actuator replacements and backwashing tests to identify issues with different Safi units either within the previous PLC's program or with the electrical equipment itself.

I was also involved with the redesign of the current Safi system to reduce expenses by limiting the number of electrical actuators and valves required. I wrote simplified programs for those systems as well.

Conclusion:

The tasks that I undertook at Safi Manzi Water was a great opportunity to practically implement my knowledge gained from previous courses at UCT, along with gathering a new found knowledge for PLC systems and different automotive products that can perform industrial or at home tasks to simplify the life of the buyer. I am still continuing to work with Safi Manzi Water within the PLC solutions project and will now be studying the programming of Siemans products to further develop the simplicity and cost effectiveness of the Safi automated unit.

Appendix A

```
from datetime import datetime, time
import RPi.GPIO as GPIO

# Set up GPIO pin
green = 17
red1 = 27
blue1 = 22
red2 = 24
blue2 = 23

GPIO.setmode(GPIO.BCM)
GPIO.setup(green, GPIO.OUT)
GPIO.setup(red1, GPIO.OUT)
GPIO.setup(blue1, GPIO.OUT)
GPIO.setup(red2, GPIO.OUT)
GPIO.setup(blue2, GPIO.OUT)

# Define the start and end times
start_time1 = time(12, 44, 10)
end_time1 = time(12, 44, 20)

start_time2 = time(12, 44, 20)
end_time2 = time(12, 44, 30)

start_time3 = time(12, 44, 30)
end_time3 = time(12, 44, 40)

start_time4 = time(12, 44, 40)
end_time4 = time(12, 44, 50)

while True:
    current_time = datetime.now().time()

    # Print the current time
```

```
print(f"Current Time: {current_time}")


# Check if the current time is within the specified interval
if start_time1 <= current_time <= end_time1:
    GPIO.output(green, GPIO.LOW)
    GPIO.output(red1, GPIO.LOW)
    GPIO.output(blue1, GPIO.HIGH)
    GPIO.output(red2, GPIO.HIGH)
    GPIO.output(blue2, GPIO.HIGH)
elif start_time2 <= current_time <= end_time2:
    GPIO.output(green, GPIO.LOW)
    GPIO.output(red1, GPIO.HIGH)
    GPIO.output(blue1, GPIO.LOW)
    GPIO.output(red2, GPIO.HIGH)
    GPIO.output(blue2, GPIO.HIGH)
elif start_time3 <= current_time <= end_time3:
    GPIO.output(green, GPIO.LOW)
    GPIO.output(red1, GPIO.HIGH)
    GPIO.output(blue1, GPIO.HIGH)
    GPIO.output(red2, GPIO.HIGH)`
    GPIO.output(blue2, GPIO.LOW)
elif start_time4 <= current_time <= end_time4:
    GPIO.output(green, GPIO.LOW)
    GPIO.output(red1, GPIO.HIGH)
    GPIO.output(blue1, GPIO.HIGH)
    GPIO.output(red2, GPIO.LOW)
    GPIO.output(blue2, GPIO.HIGH)
else:
    GPIO.output(green, GPIO.HIGH)
    GPIO.output(red1, GPIO.HIGH)
    GPIO.output(blue1, GPIO.HIGH)
    GPIO.output(red2, GPIO.HIGH)
    GPIO.output(blue2, GPIO.HIGH)
```

Appendix B

```
// CONNECTIONS:
// DS1302 CLK/SCLK --> 5
// DS1302 DAT/IO --> 4
// DS1302 RST/CE --> 2
// DS1302 VCC --> 3.3v - 5v
// DS1302 GND --> GND

#include <ThreeWire.h>
#include <RtcDS1302.h>
int led1 = 6;         // the PWM pin the LED is attached to
int led2 = 7;
int led3 = 8;
int led4 = 9;
int led5 = 10;
int dayOfWeek;
ThreeWire myWire(4,5,2); // IO, SCLK, CE
RtcDS1302<ThreeWire> Rtc(myWire);

void setup ()
{
    Serial.begin(9600);
      pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(led4, OUTPUT);
  pinMode(led5, OUTPUT);
    Serial.print("compiled: ");
    Serial.print(__DATE__);
    Serial.println(__TIME__);

    Rtc.Begin();

    RtcDateTime compiled = RtcDateTime(__DATE__, __TIME__);
    printDateTime(compiled);
    Serial.println();

    if (!Rtc.IsDateTimeValid())
    {
        // Common Causes:
        //    1) first time you ran and the device wasn't running yet
        //    2) the battery on the device is low or even missing

        Serial.println("RTC lost confidence in the DateTime!");
        Rtc.SetDateTime(compiled);
    }
```

```
    if (Rtc.GetIsWriteProtected())
    {
        Serial.println("RTC was write protected, enabling writing now");
        Rtc.SetIsWriteProtected(false);
    }

    if (!Rtc.GetIsRunning())
    {
        Serial.println("RTC was not actively running, starting now");
        Rtc.SetIsRunning(true);
    }

    RtcDateTime now = Rtc.GetDateTime();
    if (now < compiled)
    {
        Serial.println("RTC is older than compile time!  (Updating DateTime)");
        Rtc.SetDateTime(compiled);
    }
    else if (now > compiled)
    {
        Serial.println("RTC is newer than compile time. (this is expected)");
    }
    else if (now == compiled)
    {
        Serial.println("RTC is the same as compile time! (not expected but all is fine)");
    }
}

void loop ()
{
    RtcDateTime now = Rtc.GetDateTime();
    printDateTime(now);
    Serial.println();

    if (!now.IsValid())
    {
        // Common Causes:
        //    1) the battery on the device is low or even missing and the power line was disconnected
        Serial.println("RTC lost confidence in the DateTime!");
    }

    delay(1000); // five seconds
}

#define countof(a) (sizeof(a) / sizeof(a[0]))

void printDateTime(const RtcDateTime& dt)
```

```
{
    char datestring[25];
    snprintf_P(datestring,
            countof(datestring),
            PSTR("%01u %02u/%02u/%04u %02u:%02u:%02u"),
            dt.DayOfWeek(),
            dt.Month(),
            dt.Day(),
            dt.Year(),
            dt.Hour(),
            dt.Minute(),
            dt.Second() );
    Serial.println(datestring);
     if (dt.Hour()== 14 && dt.Minute()>=44 && dt.Minute()<46 && dt.Second()> 00   )
     {
      digitalWrite(led1, LOW);
  digitalWrite(led2, LOW);
   delay(30000);
  digitalWrite(led2, HIGH);
  digitalWrite(led3, LOW);
  delay(30000);
  digitalWrite(led3, HIGH);
  digitalWrite(led4, LOW);
  delay(30000);
  digitalWrite(led4, HIGH);
  digitalWrite(led5, LOW);
  delay(30000);

  } else{
    digitalWrite(led1, HIGH);
  digitalWrite(led2, HIGH);
  digitalWrite(led3, HIGH);
  digitalWrite(led4, HIGH);
  digitalWrite(led5, HIGH);

  }
}
```