Istituto Istruzione Superiore Fossano

G.Vallauri

PNEUMATIC BENCH

Customer: Gambone Fabrizio, Milanesio Mario

Speaker: Gentile Paolo, Millone Matteo, Mhilli Giulio

Istituto Istruzione Superiore "G.Vallauri" Fossano

School Year 2020/2021

Summary

INTRODUCTION

In this technical report, we are going to present and analyse a pneumatic bench managed remotely using a “Wemos” device and the “Blynk” app. The components that we used, the planing and implementation phases of the project with relative images, and at the end some final considerations will be reported below.

CONTENT ABOUT THE REPORT

The exercise is focused about the management of the control logic of an electropneumatic system using Wemos micro controller with acquisition of the relative operating data. Then the same data will be saved on a SD card

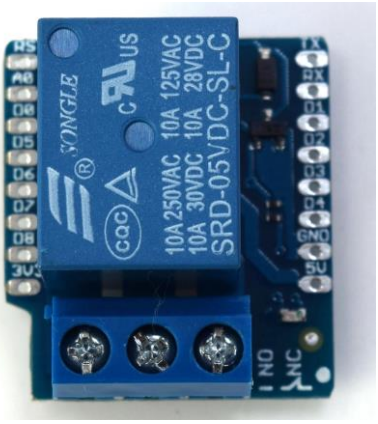
COMPONENTS

- Printed circuit board: micro controller Wemos D1 mini;

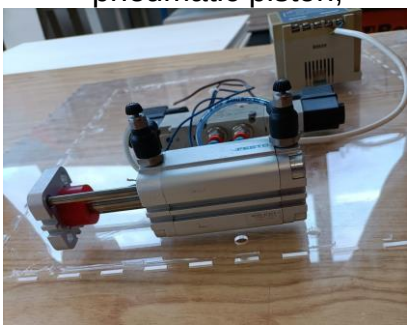
- RGB-Shield;

- Micro SD Shield;

- Relay shield;



- pneumatic piston;

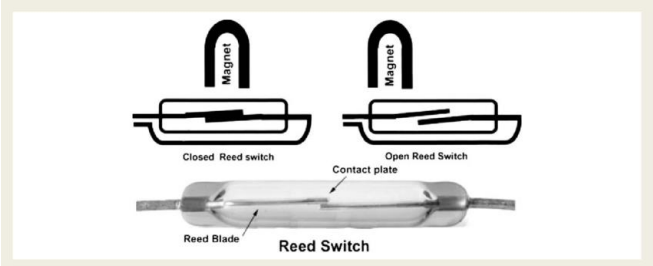


- feeder (24 V);

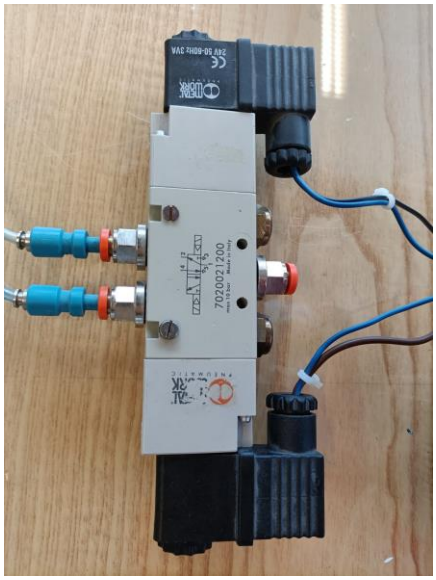
- wires;

- wooden slab;

- hot glue;
- 2 sensors of “end of the line”;



- Electrovalve



DESCRIPTION OF THE SOLUTION ADOPTED

As the teacher said, we needed to modernise the pneumatic benchmark, and the first thing was the replace of the old setting, that was a manual setting, to an automatic setting, that was able to be remotely controlled with Blynk, by using a data sheet that we have written. Moreover the old wooden support has been replaced by a new wooden box, designed and built at school with a Laser Cut.

DESCRIPTION OF THE SOLUTION WAY

- Put in writing of the data sheet;
- Configuration and setting of Blink;

THE CODE

- First paragraph contains all the libraries that are used in the sketch.

```
#include <Adafruit_NeoPixel.h>
#include <NTPClient_Generic.h>
#include <WiFiUdp.h>
#include <SPI.h>
#include <SD.h>
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
```

- In the second one, there are many function that have the task to set data and initialize components; in order: initialization of Blynk Bot, collection of Wi-Fi credentials, setting of the time zone, definition of RGB LED, how many LEDs are on the board and with which pin the system can control them, connection to the time source and, at the last raw, the initialization of the Blynk timer.

```
#define BLYNK_PRINT Serial
char auth[] = "xxxxxxxxxxx"; // Blynk Bot Auth Token
char ssid[] = "xxxxxxxxxxx"; // Your WiFi credentials.
char password[] = "xxxxxxxxxxx"; // Set password to "" for open networks.
const int chipSelect = D4;
const long utcOffsetInSeconds = 3600; // Offset in Seconds of time in your country
String weekDays[7] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};
String months[12] = {"January", "February", "March", "April",... };

#define PIN   D8 //RGB
#define LED_NUM 7 //RGB

Adafruit_NeoPixel leds = Adafruit_NeoPixel(LED_NUM, PIN, NEO_GRB + NEO_KHZ800); //RGB

WiFiUDP ntpUDP;// Define NTP Client to get time
NTPClient timeClient(ntpUDP, "pool.ntp.org");
BlynkTimer timer; // Initialize a timer by Blynk app
```

- Than there are many variables. These are the main variables in the sketch, they can be easily changed in this part:
  - StrokesBlynk = number of strokes, they are set on the Blynk Bot;
  - InitialDelayBlink = delay between the start and the first stroke in seconds, it is set on the Blynk Bot;
  - PauseBlynk = delay between two strokes;
  - Aplus = pin attached to the reed sensor that corresponds to A+, the extended position of the piston;

- Amenus = pin attached to the reed sensor that corresponds to A-, the retired position of the piston;
- Vbutton = pin attached to the Blynk virtual button, it isn't actually wired;
- relay = pin attached to the relay that controls the electrovalve which controls the piston;
- Strokes = number of strokes that piston has done,
- trueStrokes, truePause = support variables for avoiding the changing of setting on Blynk app.

```
33 int StrokesBlynk = 0;
34 int InitialDelayBlynk = 0;
35 int PauseBlynk = 0;
36
37 int Aplus = D2;
38 int Amenus = D3;
39
40 int VButton = D0;
41 int relay = D1;
42 int Strokes = 0;
43
44 int trueStrokes = 0;
45 int truePause = 0;
```

- In the setup section there are functions that have to be executed just one time:
  - leds.begin() = initializes the RGB LED;
  - Serial.begin() = initializes a type of serial monitor, in this case 9600 one;
  - Blynk.begin() = initializes the Blynk Bot, through authorization token, Wi-Fi address and its password;
  - pinMode() = sets the mode of every main pin on the board;
  - timer.setInterval() = creates a function called "finecorsa" that starts every 300 milliseconds;
  - from row 57 to row 61 there are some functions to connect the board with the SD card;
  - from row 62 to row 68 there are some function to connect the board to the Wi-Fi net.
  - Last two lines are related to the time zone offset.

```
47 void setup()
48 {
49   leds.begin(); //RGB
50   Serial.begin(9600);
51   Blynk.begin(auth, ssid, password);
52   pinMode(VButton, INPUT);
53   pinMode(relay, OUTPUT);
54   pinMode(Aplus, INPUT);
55   pinMode(Amenus, INPUT);
56   timer.setInterval(300L, finecorsa);
57   Serial.print("Initializing SD card...");
58   if (!SD.begin(chipSelect)) {
59     Serial.println("Card failed, or not present");
60     return;
61   }
62   Serial.print("Connecting to ");
63   Serial.println(ssid);
64   WiFi.begin(ssid, password);
65   while ( WiFi.status() != WL_CONNECTED ) {
66     delay ( 500 );
67     Serial.print ( "." );
68   }
69
70   timeClient.begin();// Set offset time in seconds to adjust for your timezone
71   timeClient.setTimeOffset(3600);// In Italy, There is a GMT offset of +1 hour
72 }
73
```

- The next function permits the correct start of the RGB LED.

```
74 void led_set(uint8 R, uint8 G, uint8 B) {
75   for (int i = 0; i < LED_NUM; i++) {
76     leds.setPixelColor(i, leds.Color(R, G, B));
77     leds.show();
78     delay(50);
79   }
80 }
```

- Every function called BLYNK_WRITE() has the task to collect the value of respective variable, whenever this changes;
- param.asInt = transforms a char data into a Int data, because char data are more difficult to use with other type of data;
- than, the system writes on the serial monitor the read value and the correspondent name.

```
82 BLYNK_WRITE(V0) //done strokes from Blynk to Wemos
83 {
84   StrokesBlynk = param.asInt();
85   Serial.print("Strokes: ");
86   Serial.println(StrokesBlynk);
87 }
88
89 BLYNK_WRITE(V1) //initial delay from Blynk to Wemos
90 {
91   InitialDelayBlynk = param.asInt();
92   Serial.print("InitialDelay : ");
93   Serial.println(InitialDelayBlynk);
94 }
95
96 BLYNK_WRITE(V2) //Pause between 2 strokes
97 {
98   PauseBlynk = param.asInt();
99   Serial.print("Pause : ");
100  Serial.println(PauseBlynk);
```

- In the loop function, the system cycle limitlessly unless it stops. There are many fundamental code lines:
  - row 104 = creates a file called “datalog.odf” where the board can load data;
  - Blynk.run() = permits the connection with the Blynk Bot and keep the link active;
  - timer.run() = starts counting in milliseconds;
  - In the if conditions is verified the presence of two variables:
    - VButton = the virtual button on Blynk window;
    - Amenus = the reed sensor in the retired position of the piston;

Just if these two variables are verified, the first cycle can start. The system now:

- sets the variable Strokes to 0, this parameter represents the number of already done strokes in the cycle;
- moves the values taken from the Blynk App into local variables that don’t change during the cycle (rows 108-109-110);
- waits a time equal to the “InitialDelay” set on Blynk;
- sets on HIGH the relay, so the piston comes out;
- writes on SD card the number of requested strokes.

```
103 void loop() {
104   File dataFile = SD.open("datalog.odf", FILE_WRITE);
105   Blynk.run();
106   timer.run();
107   if (digitalRead(VButton) && (digitalRead(Amenus))) {
108     Strokes = 0;
109     truePause = PauseBlynk;
110     trueStrokes = StrokesBlynk;
111     delay(InitialDelayBlynk * 1000);
112     digitalWrite(relay, HIGH);
113     dataFile.print(" requested stroke: ");
114     dataFile.println(trueStrokes);
115     dataFile.close();
116   }
117 }
```

- The “finecorsa” function is activated every 300 milliseconds. Every time it starts running, it makes some verifications:
  - if the system receive a signal from the reed correspondent to A+ (extended position), it has to:
    - set the relay on LOW, so the piston will come back;
    - set the RGB LED on red colour;
    - add 1 strokes to the previous already done strokes and write this number on Serial monitor, SD card and on Blynk App;
    - read and write on SD and serial monitor, two values: the EpochTime, that is an amount of seconds, between one defined date in the past and today, and the formatted time, that is the time like we are used to seen so as hh: mm: ss;

```
119 void finecorsa () {
120   timeClient.update();  //String dataString = "";
121   File dataFile = SD.open("datalog.odf", FILE_WRITE);
122   if (digitalRead(Aplus)) {
123     digitalWrite(relay, LOW);
124     led_set(10, 0, 0); //red
125     Strokes = Strokes + 1;
126     Serial.print("done strokes: ");
127     Serial.println(Strokes);
128     Blynk.virtualWrite(V3, Strokes);
129     unsigned long epochTime = timeClient.getEpochTime();
130     Serial.print("Epoch Time: ");
131     Serial.println(epochTime);
132     dataFile.print("Epoch Time: ");
133     dataFile.print(epochTime);
134     dataFile.print("; ");
135     dataFile.print(timeClient.getFormattedTime());
136     dataFile.print("; ");
137     dataFile.print("done strokes: ");
138     dataFile.print(Strokes);
139     dataFile.println(";");
140     dataFile.close();
141   }
```

- else if the system receive a signal from the reed correspondent to A- (retired position), it has to do a successive verification: it has to verify if the number of done strokes is higher or lower than the requested strokes. If done strokes are lower than requested ones, the system has to:
  - set the RGB LED on yellow colour;
  - wait a time equal to the value Pause, defined on Blynk App;
  - set the relay on HIGH, so the piston will come out;
  - set the RGB LED on red colour again;

```
142   else if (digitalRead(Amenus) && (Strokes < trueStrokes)) {
143     led_set(10, 10, 0); //yellow
144     delay(truePause * 1000);
145     digitalWrite(relay, HIGH);
146     led_set(10, 0, 0); //red
147   }
```

- else if done strokes are higher than requested ones, it has to:
  - set the number of done strokes to 0;
  - set the number of requested strokes to 0;
  - write the vale of new strokes requested (0) on Blynk App;
  - sets the RGB LED on green.

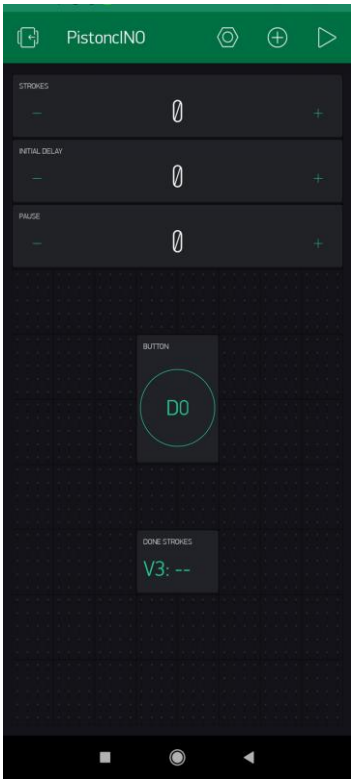
```
148   else if (digitalRead(Amenus) && (Strokes >= trueStrokes)) {
149     Strokes = 0;
150     trueStrokes = 0;
151     Blynk.virtualWrite(V3, Strokes);
152     led_set(0, 10, 0); //green
153   }
154 }
```

ATTENTION

The colour logic is:

- ↳ the LED will be red when the piston is working, it means a careful signal;
- ↳ the LED will be yellow when the piston isn't moving cause a delay, but it hasn't finished the entire cycle yet;
- ↳ the LED will be green when the piston isn't working and it's waiting for instructions for a new cycle.

CONFIGURATION AND SETTING OF BLINK



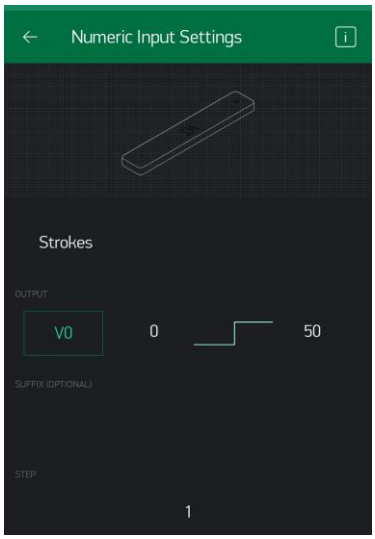
The side image represents the interface of the Blynk Bot that permits every type of communication with the board. The Bot is made by 3 main parts:

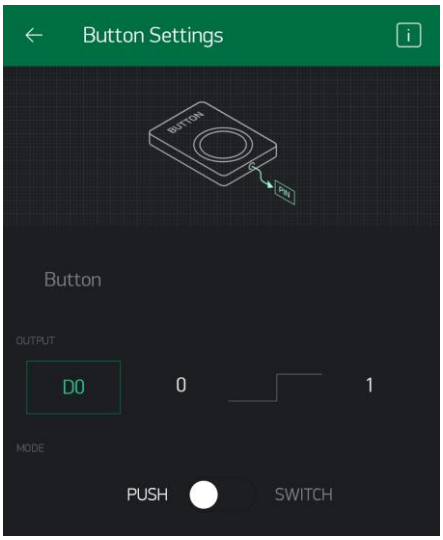
- INPUT-data part;
- Virtual Button;
- OUTPUT-data part.

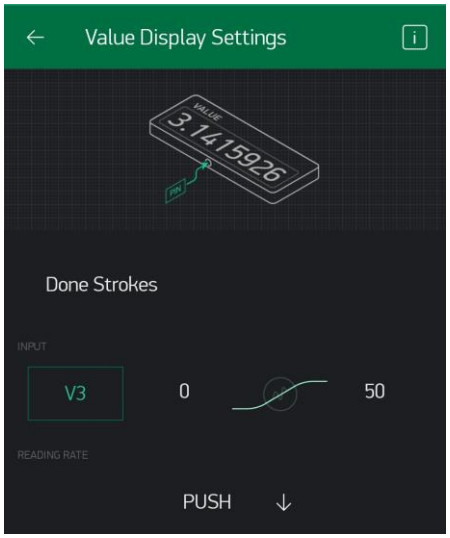
The first part is made by 3 "Numeric Input" every component corresponds to one of three main variables (Strokes, Initial Delay and Pause).

The second part is just the Button that starts the cycle and sends the data to the board.

The third part is a Value Display that corresponds to the number of done strokes.


Numeric Input Settings
Strokes
OUTPUT
V0
0
50
SUFFIX (OPTIONAL)
STEP
1


Button Settings
BUTTON
PIN
Button
OUTPUT
D0
0
1
MODE
PUSH
SWITCH


Value Display Settings
VALUE
3.1415926
PIN
Done Strokes
INPUT
V3
0
50
READING RATE
PUSH

POSSIBLE CHANGES

Despite the pneumatic bench works and the realization of all the target that the professor asked, this project could be done better. For example the datasheet: it has been written in a scholastic way, and the prof. intervene only in a few case, and it works, but it is rough, and it could be upgraded, maybe using some professional command and some better shields, and maybe using Telegram instead of Blynk, to create a different interface and using easier commands.

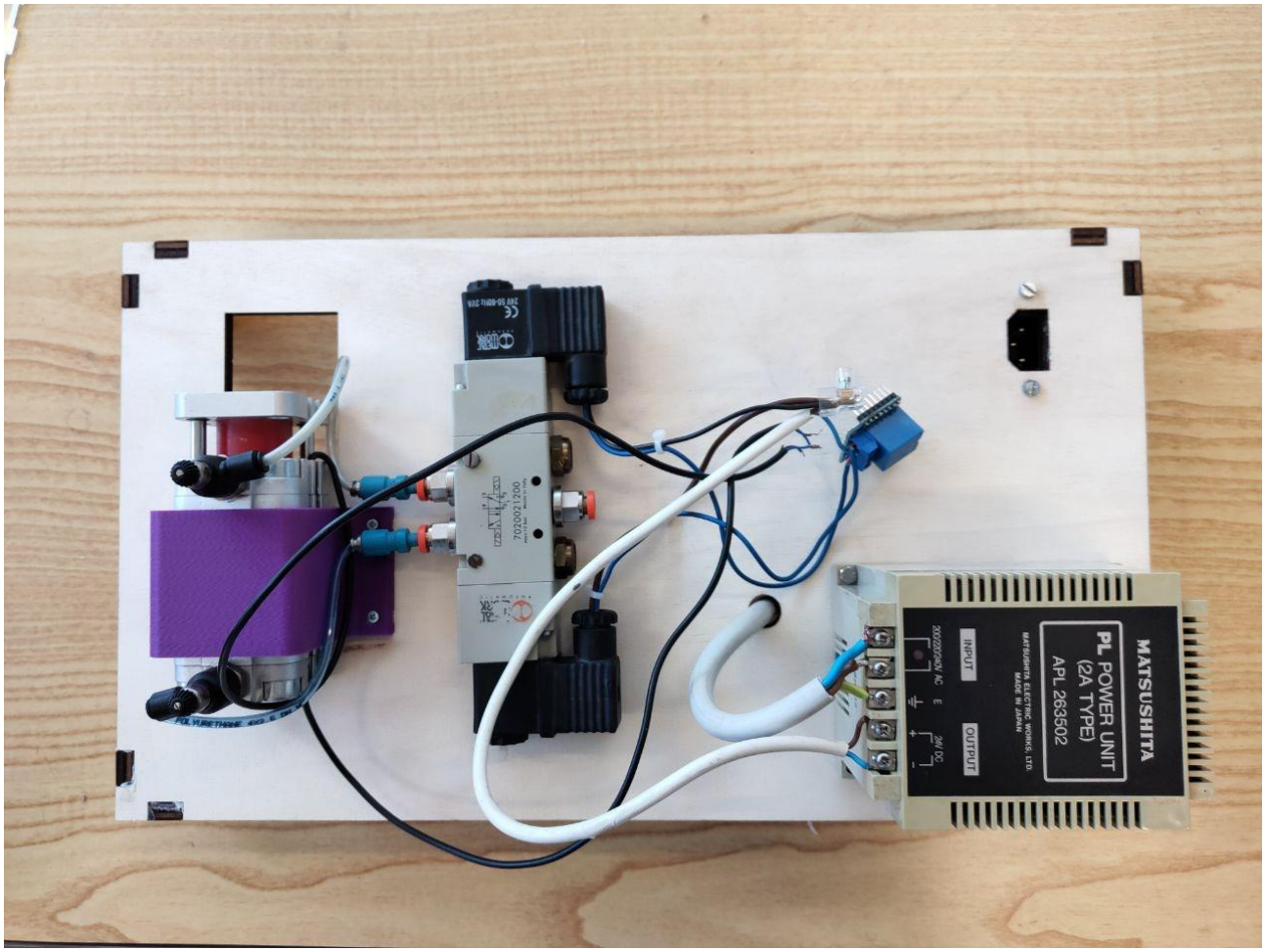
Another change that could be done to improve this datasheet is the possibility to save and load data on a cloud space, for example on Thingsboard. It is an easy platform to save data and to read constantly for example the position of the piston, the number of cycle that he has done, and even the time when he did it. It could be useful to saving data more efficiently and it could be easier to find some interruption or dysfunctions. A possible example of datasheet could be the following one:

```
#include "ThingsBoard.h"
#include <ESP8266WiFi.h>
#include <WEMOS_SHT3X.h>
SHT3X sht30(0x45);
#define WIFI_AP             "XXXXXXXXXX"
#define WIFI_PASSWORD       "XXXXXXXXXX"

// See https://thingsboard.io/docs/getting-started-guides/helloworld/
// to understand how to obtain an access token
#define TOKEN               "XXXXXXXXXXXXXXXXXXXX"
#define THINGSBOARD_SERVER  "demo.thingsboard.io"
// Initialize ThingsBoard client
WiFiClient espClient;
// Initialize ThingsBoard instance
ThingsBoard tb(espClient);
// the Wifi radio's status
int status = WL_IDLE_STATUS;
int interval = 30000;

void setup() {
  // initialize serial for debugging
  Serial.begin(115200);
  WiFi.begin(WIFI_AP, WIFI_PASSWORD);
  Serial.println("Connecting to WiFi network ...");
  // attempt to connect to WiFi network

  WiFi.begin(WIFI_AP, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connected to WiFi network");
}
void loop() {
  if (WiFi.status() != WL_CONNECTED) {
    reconnect();
  }
  if (!tb.connected()) {
    // Connect to the ThingsBoard
    Serial.print("Connecting to: ");
    Serial.print(THINGSBOARD_SERVER);
```

```
    Serial.print(" with token ");
    Serial.println(TOKEN);
    if (!tb.connect(THINGSBOARD_SERVER, TOKEN)) {
      Serial.println("Failed to connect");
      return;
    }
  }
  if (sht30.get() == 0) {
    Serial.print("Temperature in Celsius : ");
    Serial.println(sht30.cTemp);
    Serial.print("Relative Humidity : ");
    Serial.println(sht30.humidity);
    Serial.println();
  }
  else
  {
    Serial.println("Error!");
  }
  Serial.println("Sending data...");
  // Uploads new telemetry to ThingsBoard using MQTT.
  // See https://thingsboard.io/docs/reference/mqtt-api/#telemetry-upload-api
  // for more details
  tb.sendTelemetryFloat("Numero corse", 1);
  tb.sendTelemetryFloat("Temperature", sht30.cTemp);
  tb.sendTelemetryFloat("humidity", sht30.humidity);
  tb.loop();
  delay(interval);
}
void reconnect() {
  // Loop until we're reconnected
  status = WiFi.status();
  if ( status != WL_CONNECTED) {
    WiFi.begin(WIFI_AP, WIFI_PASSWORD);
    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
    }
    Serial.println("Connected to WiFi network");
  }
}
```

INTEGRATIONS

It was created a wooden box to protect and keep clean the whole system, and to avoid the movement of the components. It was realized using 8 mm thick wooden slab, and it was cut using LaserCut machine. The shape isn't much important, the main target was to save contents inside the box. Then it can be useful to move the system without many difficulties. Another hint could be to add a hole in proximity to the path of the piston, in this way it won't scratch each other. This could reduce the amount of damage.



CONCLUSIONS

This scholastic experience teaches a lot about teamwork and how much is difficult the transition from theory to practice. Even a simple box is difficult to realize, because of the existence of the holes that must be done perfectly, using a calibre; or the presence of many unexpected software problems, with Blynk app and the Wemos micro controller, that is simile to Arduino, but has got some differences in some libraries and commands.

However, the piston works, and the automatic cycle rightly end when it reaches the strokes that you can set at the beginning, and of course it wait the delay that you have setted, so professor's request have been respected. Even the LED do his work: green light when it' waiting to start, red light when it is moving, yellow light if it's waiting the setted delay. Lastly, it saves the number of strokes and the time when it happens on the SD card (if it is inserted) in a ".ods" file, so in this way it can be read by LibreOffice Calc; and it saves even on cloud.

BIBLIOGRAPHY AND ATTACHED DOCUMENTS

https://www.wemos.cc/en/latest/d1_mini_shield/index.html : the Wemos website, with all the shields about D1 mini and the technical feature of every shield. We used 4 shields and the relative examples code: RGB led, Micro SD, Relay and Tripler Base.

http://help.blynk.cc/en/articles/512061-what-is-virtual-pins: the link to understand how Blynk works, and how to read some value from Blynk to Wemos;

https://www.danielealberti.it/2017/03/tutorial-app-blynk-e-esp8266.html: this is the website that explain how to start to use Blynk and his base commands, and many hint to learn how to make the first project;

https://github.com/arduino-libraries/NTPClient: this is the github link, to use an example of connecting Wemos to a WiFi connection, and to read the hours, minutes and seconds during the execution;

https://github.com/arduino-libraries/NTPClient/blob/master/NTPClient.h: it is a reserve link, if the previous doesn't work;

https://randomnerdtutorials.com/esp8266-nodemcu-date-time-ntp-client-server-arduino/: this is the link where is explained the tutorial about how to set the real time in every timezone (like Greenwich Mean Time) , step by step, and explains how to save every type of time.

https://diyi0t.com/sd-card-arduino-esp8266-esp32/: this is a link where is a tutorial about the SD card and how to use it.

https://www.epochconverter.com/ : here it is the online converter that can change the Epoch time that is saved on the SD card, in a precious moment that is readable by a human. It changes the number of seconds between the 1/1/1970 and the moment when the data have been saved.

https://www.youtube.com/watch?…: here it is the YouTube playlist, that explains how to start using Wemos and D1 mini, uploaded by the customer of this report, and even debate about the most important shields of this board;

https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html: here it is a document that explains how does it work the ESP8266 WiFi Library, and talks about the implementation in Arduino datasheet.

https://www.arduino.cc/reference/en/: here it is the official site of Arduino Code, there are main commands for programming the board.