*Abdelkarim J. Ibreik.et.al., International Journal of Engineering Research and Applications*
*www.ijera.com*
*ISSN: 2248-9622, Vol. 15, Issue 4, April 2025, pp 64-74*

RESEARCH ARTICLE OPEN ACCESS

# Using ESP32 microcontroller in the field of Training and Industrial control process

Abdelkarim J. Ibreik * and Saad S. Alahmad **
** Eng. Abdelkarim J. Ibreik, Electrical Power Department, Public Authority for Applied Education and Training ** Eng. Saad S. Alahmad, Electrical Power Department, Public Authority for Applied Education and Training.*

**Abstract**
The importance of applying new training techniques in the field of industrial process control and instrumentation motivates trainers to find different training approaches to simplify technical concepts and attend better graduation outcomes. For these reasons, multi practical applications related to different industrial process control systems were created to enrich knowledge and improve the quality of graduated students. Therefore, a proper controller using standard input range of analogue inputs/outputs and digital inputs/outputs were used to monitor different kinds of smart sensors and control actuators. The simplicity of implementations will encourage trainees to apply more experiments and gain more experience.
This paper focused on how to use the ESP32 microcontroller with OLED in the application of industrial process using simple connections for smart sensors such as motion, fire detector and level process sensors.
The challenge was divided into two parts, firstly software, how to write the codes, gathering the libraries and programing software to monitor more than one process in a small controller with small OLED screen.
Secondly, how to create a simple shield board to fix your components and devices on it.
This board should be self-tested with built is DI (Digital Input), DO (Digital Output), AI (Analog Input), AO (Analog Output) to test the program before connecting any sensor.
**Keywords:** Industrial Control, Training Process, Control systems, microcontrollers, Sensors, Actuators, Kuwait Public Authority for Applied Education and Training.

---

Date of Submission: 08-04-2025 Date of acceptance: 19-04-2025

---

## I. Introduction

Smart Control systems are witnessing a remarkable development with the advancement of modern technology, as it has become necessary to use smart technologies to improve the performance of process control and meet the development of industrial growth. [4]

The spread of microcontrollers, which are used by hobbies to create simple applications and projects guide us to start thinking about how to use and get benefit from these controllers in the field of training in industrial control systems and building automation systems. Besides that, the availability of standard smart sensors with different functions makes the development of small control systems fast and easy.

The design in divided into 4 section, **firstly** gathering the required hardware components, **secondly** programming the software into microcontroller using ESP32 WROOM with OLED and Arduino IDE platform, **thirdly** connecting the smart sensors into the shield which designed for this purpose and **finally** compile and run the software to execute different scenarios, moreover meeting the required purpose of this research and verify that this small OLED can monitor and control the LEVEL Process along with different process at the same time. An alarms and output control can be managed automatically to control the process variable.

The importance of downloading proper libraries which are needed for programming purposes to identify and connect the output devices OLED , LCD and different applications.

The pin configuration of ESP32 is important for connections. to decide how to make your connections for input output ports , the same is shown in the Figure no.1

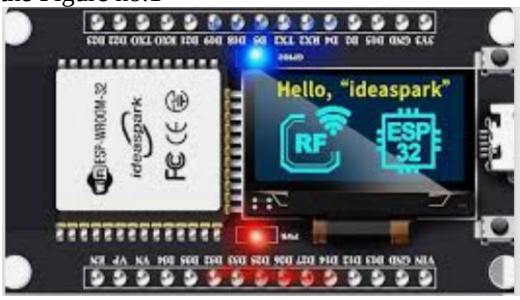


Fig.1:Photo of ESP32 with OLED and pin coding

There are different functions between the GPIO pins which enable you to choose and connect your analogue and digital inputs and outputs of your experiment. See figure No.2

By Recognize the function of each pin and the guiding libraries, The trainees will be able to understand the techniques of creating simple control circuits using small controllers with the shield designed for this purpose.

the main purpose of using ESP32 controller is summarized by:

1- Differentiate between different kinds of controller input output pins functions for future connection of processes.
2- Hands-on close loop control system with basic connections.
3- First step toward Understanding the process control, operation and connection of large-scale control systems used t in the field of instrumentation.
4- The ability to recognize and solve the troubles by simple way of tracking the faults.

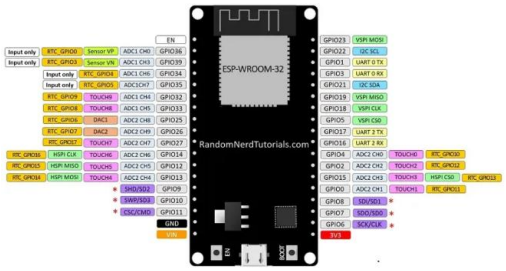



Fig.2: ESP32 pin configuration

## II. Literature Review

Any control system consists of three major components which are input, output and process, whereas the output is considered the most important part in the control systems to complete the control system.

Four major items need to be fulfilled to complete the control process:

1- Setpoint value: where the operator set the desired value of the process to maintain. For example, adjusting the required room temperature to 20 degrees centigrade.
2- Measuring process values continuously.
3- Comparing both Setpoint value with the Measured value to get the Process Error.
4- Execute the proper command output to correct the error.

### 2.1 Control Systems

Measuring process output is considered one of the most important parts for the success of any control system, therefore, precise sensors with accurate readings is preferable. Accuracy, operation range of sensors along with installation media play an important role in deciding the correct type of sensor you need to use.

Some measurements you need to monitor their results only for information and take necessary action such as weather forecast, measuring speed of winds, pressure, humidity and temperature.

other measurements you need to use in controlling your process, for example the cooling/heating system in our houses. In this type of system, we use a close loop control system with negative feedback of the measured variable. In general, the result of the process is going to be compared with the setpoint as explained before.

Control systems can be classified into two types: open loop and close loop control systems, each kind having its own application and features. Some applications are used for monitoring purposes only, by updating the status of process variables by continues measurement such as monitoring Room Temperature while heater is ON. Other applications use measures to control the Room Temperature not to go beyond the limits and keep warm weather in winter.

The thermostat was used for sensation purposes and controlling the process, while no adays microcontrollers are used along with electronic semiconductors sensors to accurate results.

#### 2.1.1 Open-Loop Control Systems

The open loop control system block diagram shows that the output-controlled variable has a measurable results, not used as feedback to close the loop, where it was applicable for controlling the process see Figure. No.3

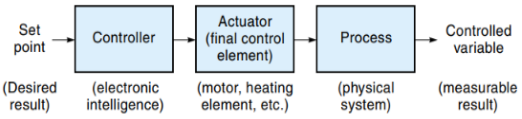



Fig.3: A block diagram of control system

This kind of control is considered simple and easy for some applications where we don't need much to worry about the deviation in the output-controlled variable with desired value in the output. Normally this kind of control needs great attention from operators and in some cases considered dangerous, moreover waste energy without getting benefit from process.

In the other hand, the close loop control system takes the measurable results as an indication for the success of process completion. Using continuous monitoring for the output variable and

comparing this information with a reference value (set point) for figure out the error of the process. See figure no. 4.

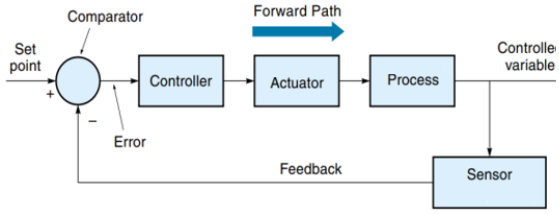



Fig.4: close loop control system

### 2.1.2 Closed-Loop Control Systems

In the closed loop systems, the measured variable will be transmitted to the process controller to calculate resultant error, a correction for the output will be taken as an order or command for actuators to complete the control action, for example controlling the servo-valves or solenoid valves. See figure no.4

### 2.1.3 Feedback Control theory

In general, the control law of Feedback is defined by the relationship between the actual measured value and desired value of a process parameter. See figure no.5

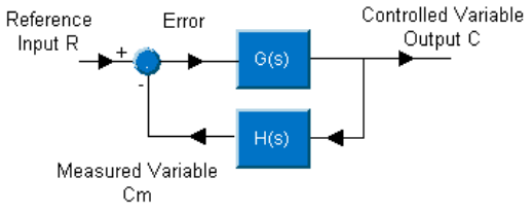



Fig. 5: General Feedback Diagram

### 2.1.4 Transfer Functions

A transfer function (TF): is considered as a mathematical relationship between the input and output of a control system component, expressed as

$$TF = \frac{output}{input}$$

Transfer Function describe the condition of the system in time-dependent and steady state and both have different characteristics, while TF at steady state is called Gain expressed as

$$TF_{steady\text{-}state} = gain = \frac{steady\text{-}state\ output}{steady\text{-}state\ input}$$

There are two periods in control systems, the first one is called Transient period where Process variable still not reach the constant value, such as a motor example a huge current needs to start up (Transient period) after certain period move to regular steady current (Steady-State period).

The potentiometer is a simple example for the TF and is used in different applications such as level the potentiometer of 10K ohms used with constant DC supply of 3.3 V regulated by the microcontroller board for this purpose. Accordingly, the percentage of the level will be converted into voltage and this value will be converted into digital number ADC . finally, we figure out how may volt need for each 1% of level.

$$TF = \frac{output}{input} = \frac{3.3\ V}{100} = 0.0333\ V/\ 1\%$$

The output can be calculated directly from TF, for example at 100%

Output= TF x Input= 0.0333V/1% x100% =3.3 V.

In this case of study, the sensor output (transmitter) will use the range 0-3.3V and not 4-20mA (0.4V to 2.0V) as used in actual industrial standard range, while the input of the level varies from 0-100%

### 2.1.5 Analog and digital Pinout

Microprocessors have different types of pins function; these pins can be used either for inputs or outputs. Or can be used for analogue or digitals signals.

Figure no. 6 shows the importance of using both types of conversion in control systems.

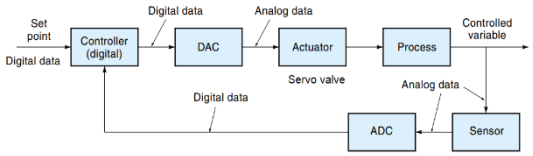



Fig.6: Using ADC and DAC with digital controller

## 2.2 Introduction to ESP32

ESP32 is a system on a chip that integrates the following features:

1. Wi-Fi (2.4 GHz band)
2. Bluetooth.
3. Dual high performance Xtensa® 32-bit LX6 CPU cores.
4. Ultra Low Power co-processor.
5. Multiple peripherals

The ESP32 peripherals include:

I. 18 Analog-to-Digital Converter (ADC) channels
II. 3 SPI interfaces
III. 3 UART interfaces
IV. 2 I2C interfaces
V. 16 PWM output channels
VI. 2 Digital-to-Analog Converters (DAC)
VII. 2 I2S interfaces
VIII. 10 Capacitive sensing GPIOs

The ADC (analogue to digital converter) and DAC (digital to analogue converter) features are assigned to specific static pins. However, you can decide which pins are UART, I2C, SPI, PWM, etc – you just need to assign them in the code. This is possible due to the ESP32 chip's multiplexing feature.[5]

The detailed pin configuration of the ESP32 is explained in figure no.7

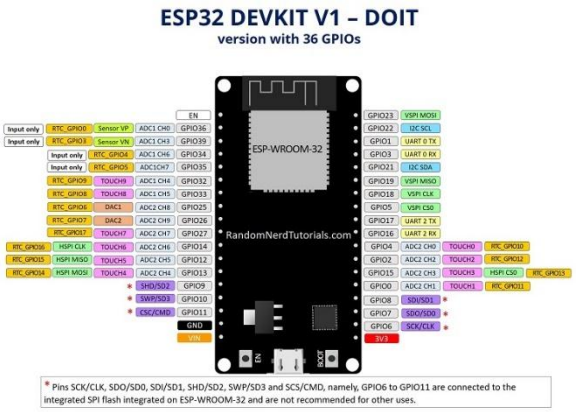



Fig.7: pin configuration of ESP32

### 2.2.1 Definitions Terminologies used in ESP32

ESP32: is a series of low-cost, low-power system-on-chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth capabilities. Developed by **Espressif Systems**.

RTC : Real Time clock
GPIO: General-purpose Input/Output
RTC-GPIO: Real Time clock-GPIO
HSPI CLOCK: Clock signal in High-Speed Serial Peripheral Interface (HSPI) communication.

ADC: Analog to Digital Converter
DAC: Digital to Analog Converter
UART: universal asynchronous receiver / transmitter. See figure no.8

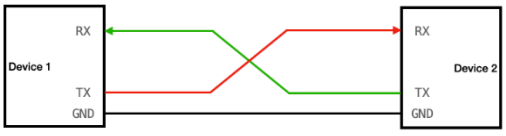



Fig.8 : UART connection diagram

UART TX: UART Transmitter
UART RX: UART Receiver
I2C: I²C means Inter Integrated Circuit (it's pronounced I-squared-C), and it is a synchronous, multi-master, multi-slave, half-duplex communication protocol. I2C uses two bidirectional open-drain lines: serial data line (SDA) and serial clock line (SCL), pulled up by resistors. You can connect: multiple slaves to one master: for example, your ESP32 reads from a BME280 sensor using I2C and writes the sensor readings in an I2C . Typically, an I2C slave device has a 7-bit address or 10-bit address. ESP32 supports both I2C Standard-mode (Sm) and Fast-mode (Fm) which can go up to 100 kHz and 400 kHz respectively
SPI: commonly used in computers and embedded systems to facilitate short-distance communication between a microcontroller and one or more peripheral integrated circuits (ICs) see figure no. 9


Fig.9: connection diagram between SPI controller and peripherals

PWM: Pulse Width Modulation, A method of encoding information based on variations of the duration of carrier pulses. Also called pulse duration modulation (PDM).

### 2.3 Signaling and signal conditioning

The signaling depend on the sensor transmitter or actuator type, Some of them use I2C protocol others use SPI. Whereas some inputs used Analogue values in the range from 0V -3.3V or Digital values.
In some cases, we need to use signal amplifier or converter such as
1- Current to pressure converters (I/P)
2- Pressure to current converters (P/I)
3- Current to voltage converter (I/V)

### 2.4 Input devices (Sensors and Transmitters)

The sensing device converts a physical quantity (e.g level, flow rate, pressure, etc.) into electrical voltage and current signal.
There are a lot of sensor types that you can use for your research or project of Arduino / esp32 controllers. To get the required measurements from Sensors we must verify the following:
1- The connection of the PINs were done in the correct location for the signal output and the power supply.
2- Install the necessary library for that sensor in software

3- Define pin connected to that sensor which will be used as input signal
4- Display results on the screen the way you find suitable for your case and write the proper commands.

If we take a simple sensor with simple Winston bridge such as the float sensor with variable resistor
The sensing device will convert the physical movement of level into electrical signal. and by increasing the level of liquid, the electrical current will be directly proportional to the rotation of potentiometer (fixed in the pulley), and the pulley. see illustrated diagram in. Figure no.10

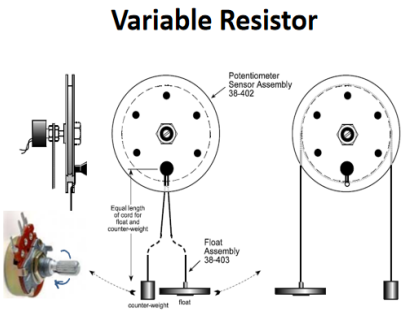



Fig.10: Potentiometer sensor assembly

**2.6 Controller**
The controller is a microprocess based device that handles reading the data from input devices and release commands to control the process, the most important part of using the controller is how to translate your thoughts into command lines, write in c-language and compile to load it to the controller
You need to use IDE arduino-1.5.8 platform.[6]

## III. The Methodology

**Introduction**
Before starting to implement this case of study, we gather the devices and components that we want to use in this experiment. Such as buying ESP32 controller and resistors, variable resistors, connection wires, LEDs, buzzer and breadboard.
The computer was prepared to upload the designed software to the controller by installing the ide Arduino. The research execution was done different steps to complete the research requirement.

**The procedure of execution**

The research has dual aims
1- Execution of practical monitoring and control of process using ESP32 with OLED
2- Training students in how to use simple components in industrial process control.

Steps of execution are summarized by the following steps:
A- Hardware Design and Installation of hardware
B- Software Design and Testing the program
C- Finally running the control system

**A- Hardware Design**
Identify the pin configuration of ESP32 to start your plan. See Figure. No.

Upper side: D23-D22-TX0-RX0-D21-D19-D18-D5-TX2-RX2-D4-D2-D15-GND-3V3
ESP 32 WROOM IDEA SPARK
Lower side: EN-VP-VN-D34-D35-D32-D33-D25-D26-D27-D14-D12-D13-GND-VIN

Fig. : ESP32 pin configuration

Decide which pin you will connect your sensor output or component to the controller.
Lower side connections D34=S.P. (AI), D32= PVar(AI), D26= LEDBoard Process AO2, D14=PUTTON_PIN1(DI), D12= LED1(DO), D13=BZR(DO)

See figure no. 11 for the connection diagram for the project

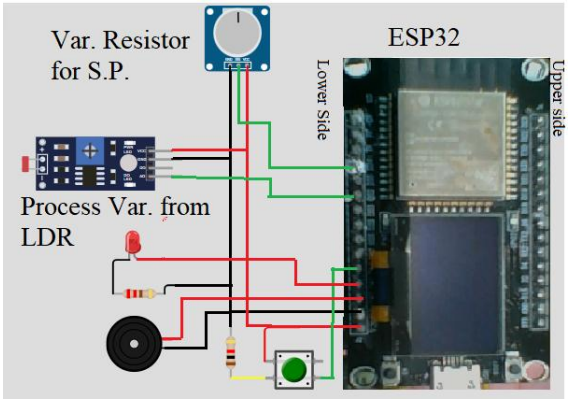



Fig. 11: connection diagram with esp32

In this figure the process value taken from LDR which give an analogue output

**B- Software Design**
1- Design the dialogue and scenario of handling the process monitoring and control.
2- Design the timing of displaying the process screen page
3- Displayed Process data on the OLED screen

Process Data
P.V.= 24%
S.P.= 50 %
Error= -26%

“ display.clearDisplay(); display.setCursor(0,0); display.display(); display.println(" Process Data"); display.print("P.V.= "); display.print(POT_OUT); display.println(" %"); display.display(); display.print("S.P.= "); display.print(SP); display.println(" %"); display.display(); float POT_Diff = POT_OUT - SP ; display.print("Diff.= "); display.print(POT_Diff); display.println(" %"); display.display(); delay(3000);

”

4- Display Process Status on the OLED screen (High Warning)

| Process status |
| --- |
| P.V→ High |

“ if (POT_OUT >SP ) { digitalWrite(PIN_LED2,HIGH);

// turn LED on:

display.println("Process Status"); display.println(" PV --> High"); display.clearDisplay(); display.setCursor(0,0); “

5- Display Process Status on the OLED screen (Low Warning)

| Process status |
| --- |
| P.V→ Low |

digitalWrite(PIN_LED2,HIGH); // turn LED on: display.println("Process Status"); display.println(" PV --> Low"); display.clearDisplay(); display.setCursor(0,0);

6- Design the SCREEN output streaming for the process value on the OLED SCREEN See figure no.12

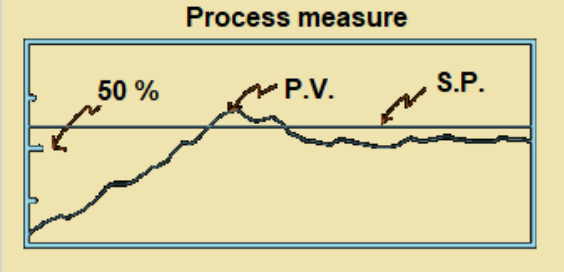



Fig.12: Real Time Streaming

“ // start Process Trends view

display.clearDisplay(); display.display(); display.setCursor(0,0); display.println("Push button for 3 sec"); display.print (" Process Trends "); display.display(); delay(3000); display.clearDisplay(); display.display();display.setCursor(0, 0); buttonState = digitalRead(BUTTON_E); while(buttonState == HIGH){ int Zi = 123;

// if (buttonState == HIGH) { digitalWrite(PIN_LED1, HIGH);

display.clearDisplay(); display.display(); display.setCursor(0,0); display.print("Process Trends-");display.print(SP); display.drawRect(3, 9,123, 23 , WHITE); // Draw Rectangle

int SP_G=SP/100*21; display.drawLine ( 3 ,30-SP_G , 123, 30-SP_G , WHITE); for (int b = 0; b < Zi; ++b) { int POT_Val = analogRead(PIN_VAR); float POT_OUT = (POT_Val * (ADC_VREF_mV / ADC_RESOLUTION) / 33 ); float R2 = POT_OUT ; Serial.println(R2); float R3=R2/100*21; display.drawLine ( 3 ,15 , 4, 15 , WHITE); display.drawLine ( 3 ,20 , 5, 20 , WHITE); display.drawLine ( 3 ,25 , 4, 25 , WHITE); // full scale 100 you can see % bars 25,50 ,75

// display.drawLine ( 3 + b , 30 , 3 + b, 30-R3 , WHITE); // full scale 100 you can see bars

display.drawLine ( 3 + b , 30-R3 , 3 + b, 30-R3 , WHITE); // full scale 100 you can see bars

display.display(); delay(100);

} buttonState = digitalRead(BUTTON_E);

}"

7- Running ide Arduino software in your computer. And connect esp32 controller to USB port, define the controller to ide software and select the connected port.
8- Program your software and select the right libraries for the simplicity of your programming. Writing the software in c language need experience and skills, in addition to that a knowledge in defining each type of numbers, variable, text is essential for format the output shape.
   For programming purposes, You need to follow the steps of writing the algorithm of the program and do some calculations. Refer to site [6.]
   There is main statements that have to be there in your programming:
   1- void setup: need to define the pins as an input or output.
   2- Void loop: need to create a loop of operation that repeated continuously.
9- Load your software after compiling to the controller and run the software to check if there is an error. Appendix[1]
10- Placing your component in breadboard, making the connections and wiring and run the controller to test your software in a simple method.
11- Finally creating your own printed circuit board, fixing your components as a final stage of project.

In this research we connect the Var. Resistor as an input for the process variable to check that the results is accurate.
It is an attractive project that makes your setup for Setpoints easy, through Var. Resistor, without the need to rewrite the value of S.P. and upload the software.

## IV. Discussions

### 4.1 Experiments and Technical works

This paper concentrates on how the trainer gain experience and improve his skills in his career, in the instrumentation field, by involving in the projects of Arduino or other controller. And this will benefit in many ways:

1- Understanding the control theories
2- Compare different sensors and actuators signals of measurements and control respectively.
3- Using the controller for Smart building or industrial control.
4- Motivate the trainees for deep study and understanding for solving hardware and software problems.

### 4.2 Experiment Results

As unexpected results might you get from the output device. These results make your work more difficult and additional efforts need to be exerted to solve the problems whether it was from the hardware or software. Fulfilling the planned scenarios and expected results will force you to have more concentration in the execution. In this research, the output results of the process appear at the oled screen, and the alarms appear at LED lights with sound alarm by Buzzer. Moreover, controlling the devices and actuators are executed by electronic switch or Relays. The control system response was tested for the purpose of applying the necessary adjustments and calibration before running, and to confirm satisfaction of the output results. This work takes time to get final and fine adjustment. Once work is completed and running, the operation of controller and applying the process will be easy.

The output of the process changed as the input of the sensor changed. The open loop control system and close loop control system can be applied in this project by simple program modifications.
The input value changes between 0 Vand 3.3 V, and the same converted into 4049 bits (Digital numbers). For example, the detected high intensity of light by LDR board produce 3.3 V. and converted by ADC to bits and displayed in Percentage 100%.
The controller was tested under different setting values and conditions. The output results were read from OLED Screen , recorded in the table and note the sound alarm (Buzzer) and Light alarm
It is to be noted that the digital output of the alarms can be occupied to control the actuators or relays to have close loop control system. Refer to Table no.1: Open Loop Control - System Response and Results

## V. Results of Discussion

Base on the steps of execution and the results which discussed in previous sections, we notice that the analogue input of any kind of process (Variable resistor used for Level measurement / Fire detection / Proximity / speed …etc.) can be adapted to the microcontroller of ESP32 by only changing the name of process. The execution of the process was simplified as much as possible to reach the target of applying industrial process by using small controllers. Following such experiments will guide the fresh trainees to create their own projects and have a better career in their future and improve the skills of tracing the hardware and software Faults and errors.

## VI. Conclusion

This paper shows that a practical part in experiments is considered a vital part for understanding the control theories which been given in lectures, realizing that there is no experiment free of faults and troubles and needs to be rectified and tuned contentiously.

The need to apply pre-defined steps for software programming is important for starting the work, but it is not enough to complete your work. Simplifying work and dividing into parts make the execution uncomplicated and basic. The research thesis should be able to explain the combined work between theories and experimental results and record the behavior of the system under different circumstances and conditions.

This research gives good knowledge about actual control systems and advice technical staff to put their hands on the practical work to fulfill their industrial projects, furthermore also applying more advanced projects using Bluetooth and Wi-Fi and sharing their knowledge and experiences with others.

## References

**Books and reference manuals**

[1.] PROCON Level, Flow & Temperature Process Control Trainers Reference Manual 38-001-3, Feedback Instrument Limited, http:www.fbk.com
[2.] PROCON Process Control Trainers Level, Flow & Temperature 38-901-M, Feedback Instrument Limited, http:www.fbk.com
[3.] Feedback equipment, Instrumentation Laboratory, Paaet, Higher Institute of Energy.

**Researches**

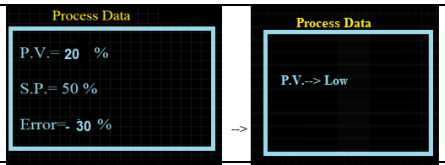
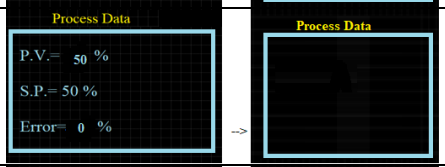
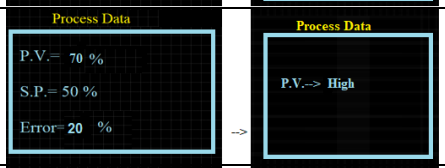
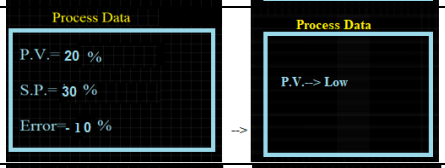
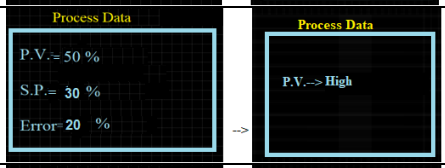
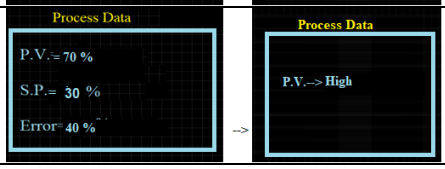
[4.] Fhaid Fahad Alrashidi., International Journal of Engineering Research and Applications www.ijera.com ISSN: 2248-9622, Vol. 15, Issue 4, April 2025, pp 10-28

**Internet**

[5.] ESP32 Pinout Reference: Which GPIO pins should you use? | Random Nerd Tutorials
[6.] Getting Started with Arduino | Arduino Documentation

**Appendix**

**Table 1: Open Loop Control - System Response and Results**

| **Results of different Setpoints and output readings**<br>Process tank used for Level measurement and control | | | | | |
|---|---|---|---|---|---|
| Case no. | Setpoint | Process Value measurement | Output Results OLED Screen | Red light Alarm LED | Sound Alarm Buzzer |
| 1 | 50% | 20% |  | OFF | ON |
| 2 | 50% | 50% |  | OFF | OFF |
| 3 | 50% | 70 |  | ON | ON |
| 4 | 30% | 20% |  | OFF | ON |
| 5 | 30% | 50% |  | ON | ON |
| 6 | 30% | 70 |  | ON | ON |
| | | | | | |

**Appendix no.1**

Programming statements used in this research

```
  // \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ Industrial Process Control  ////////////////////////////////////////////////////////////////
  //      continuous reading from process sensor (potentiometer)
// use potentiometer with variable resistor (POT.) for setting the value of S.P. = set point set
                    float SP2 = analogRead(SP1);
                    float SP= SP2/4095*100;
                    //float SP = analogRead(SP1)/4096*100;
                   display.setTextSize(0);
                   display.setCursor(0, 0);
                   display.println("  Process Measure");
```

```
// Draw Rectangle
                    display.drawRect(3, 9, 103, 15 , WHITE);
                  display.drawLine (3 +25,9 , 3 +25, 10, WHITE); display.drawLine (3 + 50,9 , 3 + 50, 11 ,
                      WHITE);
                   display.drawLine (3 +75,9 , 3 +75, 10, WHITE);
                    display.drawLine (3 +25,22 , 3 +25, 22, WHITE); display.drawLine (3 + 50,22 , 3 + 50, 21,
WHITE);
                   display.drawLine (3 +75,22 , 3 +75, 22, WHITE);
                    display.display();
                    delay(1);
                    display.println(" ");
                    display.print("  ");
                    int y = 100;
                    int POT_Val = analogRead(PIN_VAR);
                    float POT_OUT = (POT_Val * (ADC_VREF_mV / ADC_RESOLUTION) / 33 );
                   //  View the output by external 5LED Moniter
                   analogWrite(DAC_CH2, POT_OUT);
                    for (int r = 0; r <= y; ++r) {
                     if (r <= POT_OUT) {
                     display.drawLine (3 + r, 9 , 3 + r, 22, WHITE);
                     display.display();
                     delay(0);
                      } else {
                     }
                    }
                    display.println(""); display.print("     "); display.print(POT_OUT); display.println("
%");display.display();
                   delay(5000);
                    display.clearDisplay(); display.setCursor(0,0); display.display();
                     display.println(" Process Data");
                    display.print("P.V.= "); display.print(POT_OUT); display.println(" %");
                     display.display();
                     display.print("S.P.= ");  display.print(SP);  display.println(" %");
                     display.display();
                     float POT_Diff = POT_OUT - SP ;
                     display.print("Diff.= "); display.print(POT_Diff);  display.println(" %");
                     display.display();
                     delay(3000);

                      if (POT_OUT <SP ) {
                        // turn LED on:
                     digitalWrite(PIN_LED2,HIGH);
                     Serial.println("PV --> Low");  lcd.println("PV->Low");
                     display.clearDisplay();
                     display.setCursor(0,0);
                     display.println("Process Status"); display.println("  PV --> Low");
                     display.display();
                    delay(2000); // display.clearDisplay(); display.setCursor(0,0);
                    }
```

```
 if (POT_OUT >SP ) {
    // turn LED on:
 digitalWrite(PIN_LED2,HIGH);
 Serial.println("PV --> High");  lcd.println("PV->High");
 display.clearDisplay();
 display.setCursor(0,0);
 display.println("Process Status"); display.println("  PV --> High");
 display.display();
delay(2000); // display.clearDisplay(); display.setCursor(0,0);
}

 digitalWrite(PIN_LED2,LOW);

// start Process Real time  Streaming

 display.clearDisplay(); display.display(); display.setCursor(0,0);
 display.println("Push button for 3 sec");
 display.print  ("   Process Trends  ");
 display.display();
 delay(3000);
 display.clearDisplay();  display.display();display.setCursor(0, 0);
buttonState = digitalRead(BUTTON_E);
while(buttonState == HIGH){
int Zi = 123;
// if (buttonState == HIGH) {
digitalWrite(PIN_LED1, HIGH);
display.clearDisplay(); display.display(); display.setCursor(0,0);
display.print("Process Trends-");display.print(SP);
display.drawRect(3, 9,123, 23 , WHITE);       // Draw Rectangle
 int SP_G=SP/100*21;
 display.drawLine ( 3 ,30-SP_G , 123, 30-SP_G , WHITE);
for (int b = 0; b < Zi; ++b) {
 int POT_Val = analogRead(PIN_VAR);
 float POT_OUT = (POT_Val * (ADC_VREF_mV / ADC_RESOLUTION) / 33 );
 float R2 = POT_OUT ;
 Serial.println(R2);
 float R3=R2/100*21;
display.drawLine ( 3 ,15 , 4, 15 , WHITE);display.drawLine ( 3 ,20 , 5, 20 ,
WHITE);display.drawLine ( 3 ,25 , 4, 25 , WHITE);  // full scale 100   you can see % bars
25,50 ,75
//   display.drawLine ( 3 + b , 30 , 3 + b, 30-R3 , WHITE);  // full scale 100 you can see bars
   display.drawLine ( 3 + b , 30-R3 , 3 + b, 30-R3 , WHITE);  // full scale 100 you can see bars
   display.display();
   delay(100);
}
//  }else{}
   buttonState = digitalRead(BUTTON_E);
   }
   delay(1000);
display.clearDisplay(); display.setCursor(0, 0); display.display();
```