Design and Development of a Smart Health Device Based on the Internet of Things (IoT) Using ESP32

Igo Firnandes, Ilmiyati Rahmy Jasril

Department of Electronics Engineering, Universitas Negeri Padang, Padang, Indonesia

ORCID: https://orcid.org/0009-0008-3731-2143
ORCID: https://orcid.org/0009-0008-9796-7221

***Abstract* - The objective of developing the Smart Health Device utilizing the Internet of Things (IoT) with ESP32 is to offer a practical solution for users who are unable to visit the hospital for regular check-ups. This device is designed following the waterfall methodology, which encompasses requirements analysis, system design, implementation, and testing. The primary controller of this system is the ESP32 module, which is integrated with an input circuit that includes a distance sensor, weight sensor, heart rate sensor, temperature sensor, pressure sensor, and speaker. This design is capable of calculating, monitoring, and managing medical check-up values through the Blynk IoT application. The results indicate that the Smart Health Device based on IoT with ESP32 operates as intended.**

***Keywords*— Design and Development, Smart Health, NodeMCU ESP32, Internet of Things.**

I. Introduction

Health is an essential basic need for every individual, regardless of gender, age, ethnicity, or social group. When someone experiences a health disorder and is diagnosed as ill, it can lead to consequences such as the inability to work. This reason is valid as grounds for leaving duties, which in turn can result in a decrease in productivity and income for the individual or the company[1]. The communities living in Disadvantaged, Border, and Outlying Areas (DTPK) often face difficulties in accessing quality primary healthcare services.[2].

A Medical Check-Up is a comprehensive health examination process that includes various stages. It begins with consultation and medical interviews with a doctor, evaluation of health history, and examination of internal and external body organs. The primary goal is to detect health problems or diseases as early as possible. By undergoing this examination, it is hoped that any health issues can be detected early, allowing for timely action and appropriate treatment before the condition worsens.[3].

The relationship between health and Medical Check-Ups is very close, but people in Indonesia are still less concerned about undergoing these health checks. Many of them are reluctant to do so. Besides the lack of awareness of the importance of maintaining health, this is also due to the difficulty of access for people living in remote areas. The long distance between their homes and healthcare facilities makes them unwilling to undergo Medical Check-Ups. Not only rural communities, but also urban residents are hesitant to go to hospitals for these check-ups due to the time wasted as a result of long waiting times to receive Medical Check-Up services. Therefore, a solution is needed to provide more practical and efficient Medical Check-Ups, which can reduce the obstacles that people might face in undergoing health checks. To achieve this, a device is being designed that allows for faster Medical Check-Ups without requiring a direct visit to the hospital or nearby healthcare centers. This device will use Internet of Things (IoT) technology to remotely connect and monitor patients' health conditions..

Internet of Things (IoT) is the latest trend in the world of technology with the potential to become a major aspect in the future. The concept of IoT aims to expand the benefits of continuous internet connectivity by integrating physical and virtual objects through data capture and communication capabilities.[4].

This Final Project focuses on the development of a Smart Health Monitoring tool capable of performing Medical Check-Ups for several parameters, including heart rate, blood pressure, body temperature, height, and weight. All of this data will be combined to calculate the patient's Body Mass Index (BMI). The tool also has the capability to send data to the Blynk

DOI: 10.18782/ijsehtxx-xx
https://doi.org/10.18782/ijsehtxx-xx

Corresponding author: - ,
affiliation
Email: -, firnandesigo@gmail.com

Received: August 2024.
Revised: September 2024.
Accepted: October 2024.
Published: October 2024.

application, enabling patients to perform Medical Check-Ups without needing to visit the hospital. Additionally, doctors can access the patient's condition through the data sent by the tool to the Blynk application. Thus, this tool provides convenience for patients in monitoring their health independently and offers relevant information to doctors for making appropriate medical decisions.

II. LITERATUR REVIEW

A. Medical Check-Up

A Medical Check-Up is a comprehensive health examination procedure aimed at early detection of diseases or health disorders. Through this examination, it is hoped that health conditions can be detected early on. This test is also useful for planning appropriate treatment and care methods before the health condition worsens[5].

B. Heart rate

The heart is a vital organ with a heavy workload, functioning continuously as a pump that circulates blood throughout the body. The condition of the heart is influenced by diet and daily activities, and its function tends to decline with age. A normal heart rate in humans typically ranges from 60 to 100 beats per minute (bpm), and maintaining this range is important to optimize oxygen delivery throughout the body [6].

C. Body Temperature

Body temperature is the balance between heat production and heat release from the body, measured in units of heat called degrees. The temperature in question refers to the level of heat or coldness of a substance. Body temperature is the difference between the amount of heat produced by bodily processes and the amount of heat released to the external environment[7].

D. Indeks Massa Tubuh (IMT)

The weight and height of a person are usually measured using different methods and measuring tools. Additionally, calculating BMI is often done manually. This phenomenon highlights the importance of health monitoring and weight control to reduce the risk of obesity-related diseases and maintain optimal health[8].

E. Microcontroller

A microcontroller is a computer system that has one or several highly specific tasks, unlike a personal computer which has a wide range of functions[9].

F. Components Used

1) MLX90614 Temperature Sensor

The MLX90614 Temperature Sensor is a non-contact temperature sensor that measures temperature based on infrared radiation emitted by an object. This sensor can detect electromagnetic waves in the range of 700 nm to 14,000 nm and can accurately measure the internal temperature of a human up to a distance of 5 cm[10].

Picture 1. Sensor MLX90614

2) MPX5700DP pressure sensor

Pressure, denoted by the letter (p), is a physical unit that represents force per unit area (A) on a substance, such as a liquid or gas. A pressure sensor functions to convert that pressure into an induced signal or an electrical signal that can be measured and further analyzed[11].

Picture 2. MPX5700DP pressure sensor

3) Sensor MAX30100 pulse Oximeter

The MAX30100 sensor is a device that integrates pulse oximetry, heart rate monitoring, and blood oxygen content measurement. This device is equipped with two LEDs and one photodiode, and operates with a power supply of 1.8V and 3.5V.3,5V[12].

Picture 3. Sensor MAX30100

4) Sensor Ultrasonic HC-SR04

A sensor functions to provide feedback information that is useful for controlling a program by detecting the output of a particular system or environment[13].

Picture 4. Sensor Ultrasonic HC-SR04

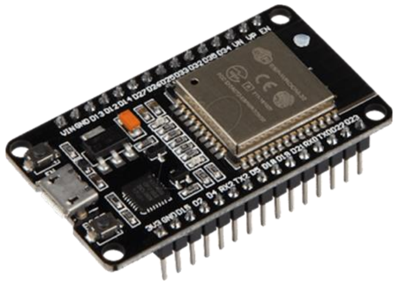
5) Sensor Load Cell

A Load Cell is a testing device for electrical equipment that functions to convert one form of energy into another. This device is commonly used to transform a force into an electrical signal[14].

Picture 5. Sensor Load Cell

6) Microcontroller Esp32

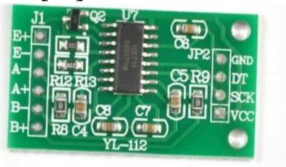
The ESP32 is a microcontroller introduced by Espressif Systems and is the successor to the ESP8266 microcontroller. The advantages of the ESP32 microcontroller compared to others include several aspects such as a greater number of pinouts, more analog pins, larger memory, the presence of Bluetooth 4.0 low energy, and WiFi capabilities. These features enable the ESP32 microcontroller to be applied in Internet of Things (IoT) applications[9].



Picture 6. Microcontroller Esp32

7) Module HX711

The HX711 module is a device that simplifies the reading of a Load Cell in weight measurements. Its function is to amplify the output signal from the Load Cell sensor and then convert the analog data into digital data that can be interfaced with a microcontroller[15].



Picture 7. Module HX711

8) Liquid Crystal Display 16X2 (LCD)

Liquid Crystal Display (LCD) is an electronic component that functions to display data, which can include characters, letters, symbols, or graphics. Due to its smaller size, LCDs are typically used with microcontrollers. LCDs are available in module form and include data pins, power supply control, and contrast adjustment.

Picture 8. LCD 20x4

9) Power Supply

A power supply, often referred to as a power source, is an electronic device used to lower voltage and convert alternating current (AC) to direct current (DC). The power supply functions as a source of power for electronic devices that require direct current (DC) to operate..

Picture 9. Power supply

10) A DC water pump

A DC water pump is a type of pump that uses a direct current (DC) motor as its power source. By applying a voltage difference across the two terminals, the motor will rotate in one direction, and if the polarity of the voltage is reversed, the direction of the motor's rotation will also reverse.

Picture 10. A DC water pump

11) Speaker

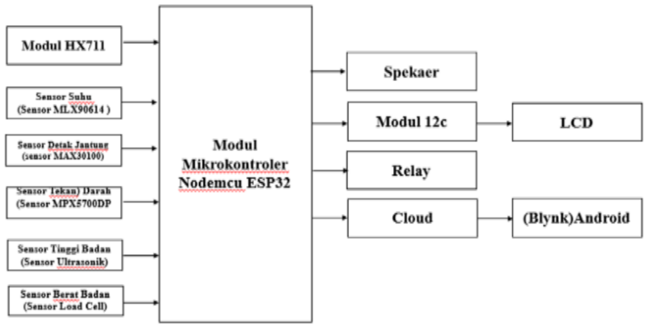
A speaker is a hardware device that functions as an output to produce sound from processed audio[16].

Picture 11. Speaker

III. METHODS

In the design of a Smart Health Tool based on the Internet of Things (IoT) using ESP32, the waterfall model is a systematic and sequential research model that is suitable for this study because it provides step-by-step procedures that align with field conditions. Although this method is considered outdated, it remains relevant for developers. The following are the procedural steps of the waterfall development method[17].

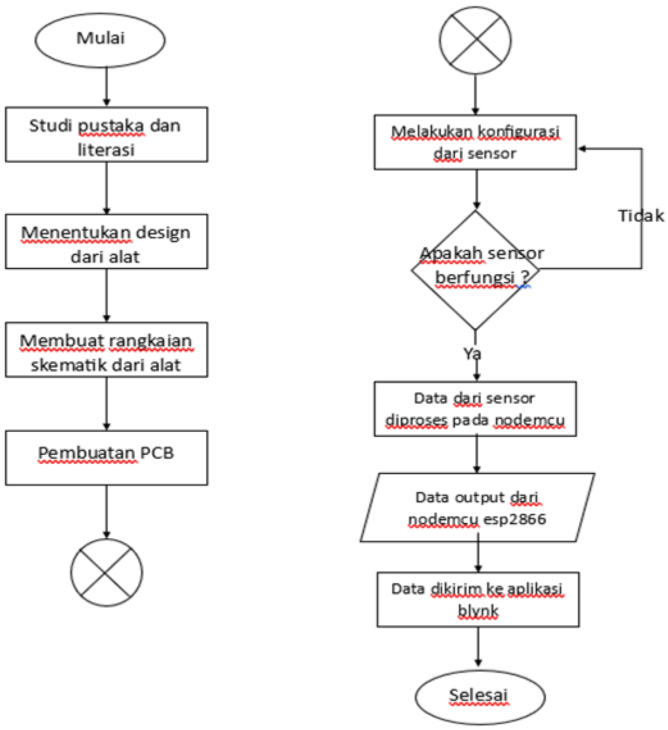
A. *Block diagram*



Picture 12. Block diagram

B. *Flowcharts Sistem*

Flowcharts are used to understand the workflow of the designed tool. The workflow of this tool is shown in the picture.

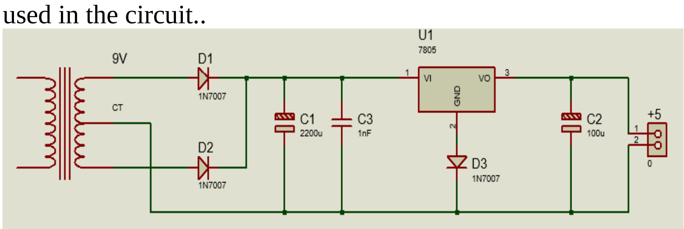


Picture 13. Flowcharts Smart Healt

C. *Hardware design*

1) Power supply design

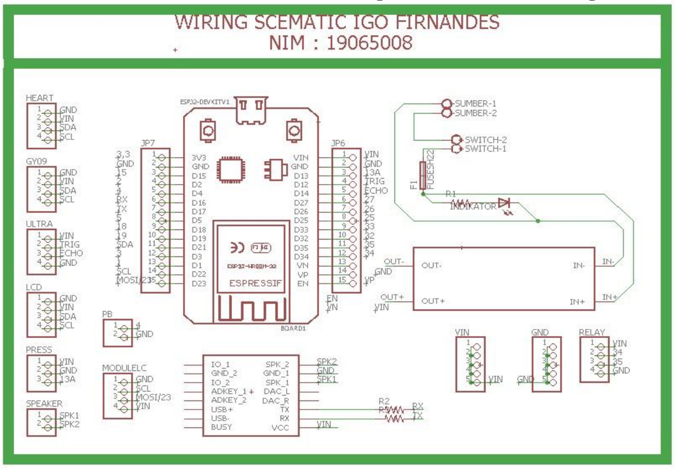
The power supply circuit functions as a source of current used in the circuit..



Picture 14. *Power supply design*

2) Overall design

Designing with temperature sensors, heart rate sensors, blood pressure sensors, ultrasonic sensors, and load cell sensors functions as input to measure the results of these sensors in real-time. All sensors require a 5V DC voltage..



Picture 15. Overall design

D. *Software design*

The software used to create the program that provides sensor information for the device includes two types of software:

1) Arduino IDE

The program to be created using the Arduino IDE is a program for the ESP32 microcontroller to connect to the Blynk application..

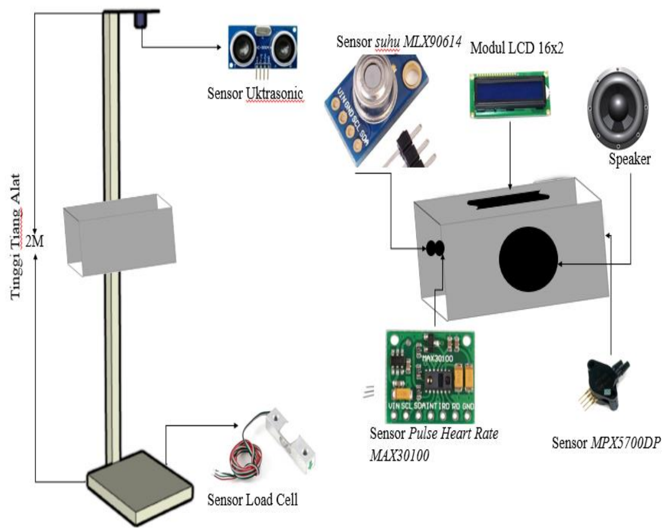
2) Blynk IoT

The Blynk IoT application is used for monitoring devices. The creation of the monitoring dashboard is done and configured directly in the Blynk IoT application.

Picture 16. Blynk IoT Application Appearance Design

E. *Mechanical design*

The materials used in the construction of this device include a thin iron beam for the base and a platform with dimensions of Length x Width (30cm x 30cm), which serves as a place to position the Load Cell for measuring body weight. Additionally, there is a 2-meter long iron beam used as a pole for measuring height. A solid object is used as a head surface barrier to ensure more accurate measurements from the ultrasonic sensor. A black box is used to house the necessary electronic equipment, along with a support for the Ultrasonic Sensor. This design aims to facilitate and ensure the availability of the materials and equipment needed for the construction of a smart health device. The design also ensures that the device will have appropriate dimensions and accuracy for measuring temperature, heart rate, blood pressure, weight, and height within certain limits.



Picture 17. Mechanical design

IV. RESULTS AND DISCUSSIONS

This chapter will discuss the testing of a smart health device based on the Internet of Things (IoT) using the ESP32. The testing includes both software testing of each system and overall system testing. After the completion of the hardware circuit assembly, the next step is to create a flowchart that facilitates communication between the running programs.

A. *Functional Testing*

Functional testing part by part based on flowcharts and the overall system consisting of :

1) Testing of HC-SR04 Ultrasonic Sensor and Load Cell: This test is conducted to determine the ability to measure body mass index. The ultrasonic and load cell sensors must be able to differentiate measurement results that are already registered in EEPROM. When the ultrasonic sensor is approached by an object, the output produced will be high, and when the object is moved away, the output will decrease. On the other hand, the load cell sensor will produce an output based on the weight of an object when pressure is applied. If both sensors are used simultaneously, the result will indicate whether the body mass index is within an ideal range or not. To determine the values received by the ESP32, the sensor detection results are sent through the serial monitor in the Arduino application. The sensor measurement results will be defined and processed by the ESP32 before sending the data to Blynk and the LCD.

Table 1. Ultrasonic Testing Results of HC-SR04

Condition of the HC-SR04 Ultrasonic Sensor	Operating Voltage of the HC-SR04 Ultrasonic Sensor 3V
Inactive	0 V
Active	3 V

```
NewPing sonar(TRIGGER_PIN, ECHO_PIN, 200);
```

```
float tinggiBadan = 192.0 - sonar.ping_cm();
```

Picture 18. Ultrasonic Program

The program is designed to periodically read distance data from an ultrasonic sensor and print it to the Serial Monitor every 500 milliseconds. The variable `newDataReady` is used to check whether new data is available. The measured distance data, stored in the `tinggi` variable in centimeters, is considered valid only if it is a non-negative value. The program then checks if the desired time interval to print the data to the Serial Monitor has been reached using the `millis()` function. If the condition is met, the program prints the measured distance information along with the unit in centimeters to the Serial Monitor and updates the last time (`t`). This process continuously monitors changes in distance from the ultrasonic sensor..

Table 2. Load Cell test results

Load Cell sensor condition	Load Cell sensor working voltage 3V
Inactive	0 V
Active	3 V

```
void loadCell(){
  Serial.println();
  Serial.println("Starting...");

  LoadCell.begin();
  LoadCell.start(2000, true);
  if (LoadCell.getTareTimeoutFlag() || LoadCell.getSignalTimeoutFlag()) {
    Serial.println("Timeout, check MCU>HX711 wiring and pin designations");
    while (1);
  } else {
    LoadCell.setCalFactor(1.0);
    Serial.println("Startup is complete");
  }
  while (!LoadCell.update());
  calibrate();
}
```

Picture 19. Load Cell Program

The program above aims to initialize and calibrate the load cell sensor using the HX711_ADC library. Initially, the program sets a default calibration value (calibrationValue = 25.41) and starts the sensor. Next, a stabilization time of 2 seconds is carried out to ensure stable values. If a timeout occurs during stabilization, the program provides an error message. If not, the calibration value is set, and the message "Startup is complete" is displayed. The program then displays several sensor parameters such as calibration value, conversion time, sampling rate, and settling time through the Serial Monitor.

```
      if (millis() - cTime1 > 3000){
        if(BMI >= 5.0 && BMI < 16){
          myDFPlayer.play(1); cTime1 = millis();
        } else if(BMI >= 16 && BMI < 18){
          myDFPlayer.play(2); cTime1 = millis();
        } else if(BMI >= 18 && BMI < 25){
          myDFPlayer.play(3); cTime1 = millis();
        } else if(BMI >= 25 && BMI < 27){
          myDFPlayer.play(4); cTime1 = millis();
        } else if(BMI >= 27){
          myDFPlayer.play(5); cTime1 = millis();
        }
      }
      Serial.println(""); cTime = millis();
    }
  }
```

Picture 20. Program BMI

In the picture, a body mass index (BMI) program is explained. Once both sensors provide their outputs, the body mass index result will be displayed. To determine the body mass index, the formula is BMI = WEIGHT / (HEIGHT)^2. After obtaining the results, the data will be sent to the Blynk application and displayed on an LCD screen.

2) Testing Pulse Heart Rate Sensor and MLX90614 Temperature Sensor: This testing is conducted to assess the capability of measuring heart rate and body temperature. The Pulse Heart Rate sensor detects the heartbeat when it is placed on the thumb, and the sensor processes this information. The MLX90614 sensor, when placed on the body, also detects the temperature and processes it. The output from the MLX90614 sensor is obtained after acquiring the heart rate data from the Pulse Heart Rate sensor. To determine the values received by the ESP32, the sensor detection results are sent via the serial monitor in the Arduino application. The sensor detection results are then defined and processed by the ESP32 before sending the data to Blynk and the LCD.

Table 3. Testing the MLX90614 Pulse Heart Rate and Temperature Sensor

Sensor	Sensor condition	Sensor working voltage
Pulse Heart Rate	Inactive	0 V
	Active	3 V
MLX90614	Inactive	0 V
	Active	3 V

```
n = n + 1;
int dataadc1 = analogRead(D0);
lcd.setCursor(0, 0);
lcd.print("n = ");
lcd.print(n);
lcd.setCursor(0, 1);
lcd.print("BPM = ");
lcd.print(bpm);
Blynk.virtualWrite(V11, n);
if ((dataadc1 > 512) && (x > 1)) {
  digitalWrite(13, HIGH);
  x = 0;
  bpm = bpm + 1;
} else if ((dataadc1 < 512) && (x < 1)) {
  x = 2;
  digitalWrite(13, LOW);
}

delay(200);
```

Picture 21. Program Pulse Heart Rate

The working principle of this program is to detect changes in sensor values to calculate heart rate, while monitoring and displaying information on the LCD and Blynk.

```
present = ds.reset();
ds.select(addr);
ds.write(0xBE);

for (i = 0; i < 9; i++) {
  data[i] = ds.read();
}

int16_t raw = (data[1] << 8) | data[0];
if (type_s) {
  raw = raw << 3;
  if (data[7] == 0x10) {
    raw = (raw & 0xFFF0) + 12 - data[6];
  }
} else {
  byte cfg = (data[4] & 0x60);
  if (cfg == 0x00) raw = raw & ~7;
  else if (cfg == 0x20) raw = raw & ~3;
  else if (cfg == 0x40) raw = raw & ~1;
}
celsius = (float)raw / 16.0;
```

Picture 22. Program MlX90614

The program above uses the OneWire library to communicate with the MLX90614 temperature sensor. First, the program performs a reset on the sensor and selects its address. Next, it sends a command to measure the temperature (0x44) and allows time for the measurement. After that, the program reads the temperature data from the sensor and combines two bytes into a single 16-bit (raw) value. The program then adjusts the raw value based on the sensor's configuration, such as shifting the raw value if the sensor has high resolution (type_s). The Celsius value is calculated from the raw value by dividing it by 16.0. This entire process is used to read the temperature data from the MLX90614 sensor and convert it into a temperature value in degrees Celsius, which is then sent to the Blynk application for display..

3) Testing of MPX5700DP Pressure Sensor: This test is conducted to determine the ability to measure blood pressure in the human body. The MPX5700DP pressure sensor captures the measurement by attaching a sphygmomanometer to the wrist, and the device will read the blood pressure gradually. Wait a few moments to obtain the final result. The output generated by the sensor and the values received by the ESP32 are sent via the serial monitor in the Arduino application. The sensor's measurement results will be defined and processed by the ESP32 before sending the data to Blynk and the LCD.

Table 4. MPX5700DP

Condition sensor MPX5700DP	The working voltage of the MPX5700DP sensor is 3V
Inactive	0 V
Active	3 V

```

int PressureValue = analogRead(PressureSensorPin);
float tekanan = (PressureValue / 4095.0) * 5000000;  // Konversi nilai ADC ke tekanan dalam Pascal
    tekanan = tekanan / 133.322; //konversi nilai pascal ke mmHg

```

Picture 23. Program Sensor MPX5700DP

4) Speaker Testing: This test is conducted to determine the quality of the sound output from the BMI. A DFPlayer MP3 module is used in the speaker to read the output from the BMI on the ESP32 and pass it to the speaker.

Table 5. Pengujian Speaker

Kondition speaker	Tegangan kerja Speaker
No Sound	0 V
Sound	3 V

```
if (!myDFPlayer.begin(Serial2, /*isACK = */true, /*doReset = */true)) {
  Serial.println(F("Unable to begin:"));
  Serial.println(F("1.Please recheck the connection!"));
  Serial.println(F("2.Please insert the SD card!"));
  while(true);
} Serial.println(F("DFPlayer Mini online."));
myDFPlayer.setTimeOut(500);
myDFPlayer.volume(DFVolume);
myDFPlayer.EQ(DFPLAYER_EQ_NORMAL);
myDFPlayer.outputDevice(DFPLAYER_DEVICE_SD);
```

Picture 24. program Df Player

The program begins with initializing serial communication through `SoftwareSerial` at a baud rate of 9600 bps to communicate with the DFPlayer Mini module. Next, the program initializes the DFPlayer Mini using the `myDFPlayer.begin(mySoftwareSerial)` method. If the initialization fails, the program prints an error message "Unable to begin DFPlayer" and halts execution with a `while (true)` statement. If successful, the program sets the timeout for the

DFPlayer Mini and adjusts the sound volume to 25. Afterward, the program pauses execution for 2 seconds before proceeding to the main part of the program, which can be filled with control logic or sound playback as needed.

5) The program begins with initializing serial communication through `SoftwareSerial` at a baud rate of 9600 bps to communicate with the DFPlayer Mini module. Next, the program initializes the DFPlayer Mini using the `myDFPlayer.begin(mySoftwareSerial)` method. If the initialization fails, the program prints an error message "Unable to begin DFPlayer" and halts execution with a `while (true)` statement. If successful, the program sets the timeout for the DFPlayer Mini and adjusts the sound volume to 25. Afterward, the program pauses execution for 2 seconds before proceeding to the main part of the program, which can be filled with control logic or sound playback as needed.

```
lcd.setCursor(0,0); lcd.print("Suhu:"); lcd.print(suhuTubuhC);
                    lcd.print("C Tensi:"); lcd.print(tekanan); lcd.print("  ");
lcd.setCursor(0,1); lcd.print("Tinggi : "); lcd.print(tinggiBadan);
                    lcd.print("Cm  ");
lcd.setCursor(0,2); lcd.print("Berat Badan :  "); lcd.print(weight);
                    lcd.print("kg  ");
lcd.setCursor(0,3); lcd.print("BMI Index :  ");  lcd.print(BMI);
                    lcd.print("  ");
```

Picture 25. Program LCD

In the picture, it can be explained that the program displays data for height, weight, BMI, pulse rate/heart rate, body temperature, and blood pressure, all read by the sensors with their respective units. This data will then be read by the Blynk application.

6) Testing ESP32 Connection with Blynk Application: After designing and creating the Blynk application as outlined in Chapter III, the Blynk application serves as a control platform for both iOS and Android operating systems to manage the ESP32. To test the Blynk application, download it from the Play Store onto your phone. Open the application and log in using the email: socong27@gmail.com. The ESP32 program will connect with the Blynk application, and Blynk will send a token code to the email. Copy and paste this token code into the Arduino IDE program as shown in the picture..

```
#define ssid       "192.168.50.10"
#define password   "12345678"
#define auth   BLYNK_AUTH_TOKEN
```

Picture 26. Program Testing ESP32 connection to Blynk

The picture can be explained that to connect to run the Blynk application you need an Auth token (chart Auth[]) which is sent via email, then connecting to the Esp32 with Blynk then turning on the Hostpot on the cellphone and WiFi on the Esp32.

7) Relay Testing for Motor Relays and Selenoids: Relay testing aims to control the motor in a series of tools. Testing is carried out manually which has been set in the Blynk application.

```
pinMode(motorRelay, OUTPUT);
pinMode(selenoid, OUTPUT);

digitalWrite(selenoid, HIGH);
digitalWrite(motorRelay, HIGH);
```

Picture 27. Motor Relay and Selenoid Program Testing

The picture above shows that when the pump status is manually selected, the pump condition is selected, then the manual water pump is equal to 0, which can be set by itself. When the pump status is on, the condition is selected to be equal to 1 and when the water pump status is off it is equal to 0, as is the solenoid. You can see the pump and solenoid test results in the table below.

Table 6. Relay test results with DC motors and selenoids

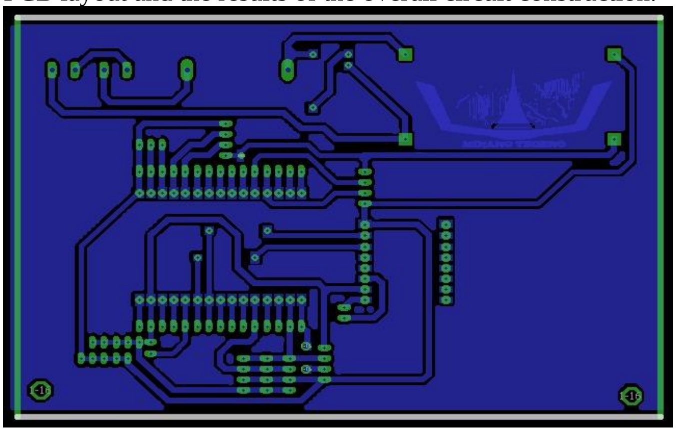
Komponen	Hihg/low	bilangan	Kondisi komponen
Motor Dc	High	1	menyala
	Low	0	mati
Selenoid	High	1	menyala
	Low	0	mati

From the relay testing results, it has been proven that the relay operates correctly, as evidenced by the relay being activated using voltage from the GPIO PIN of the NodeMCU ESP32.

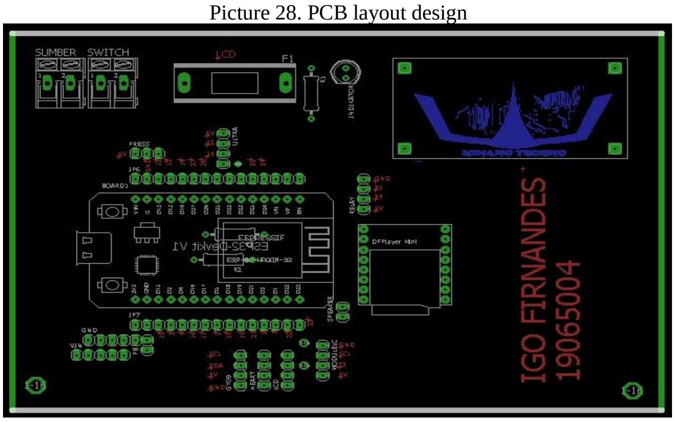
B. Tool Making Results

1) Hardware Manufacturing Results

PCB layout design is done using Eagle software because it has many libraries of electronic components. Additionally, if a component library is not available, users can also create their own component libraries. The design of the PCB layout and the results of the overall circuit construction.



Picture 28. PCB layout design



Picture 29. Overall circuit layout

2. Software Design Results

a. Widget

In designing the Blynk IoT control dashboard, the widgets or components used are :

1) Labeled Value is used for monitoring body temperature, heart rate, height, weight, BMI, and blood pressure.
2) Button is used to activate the relay, motor, and solenoid. Figure 2 shows an example of a low-resolution image which.

b. Blynk IoT display

Picture 30. Blynk IoT Devices display and control dashboard

3. Mechanical Design Results

Picture 31. Mechanical Manufacturing Results

V. CONCLUSIONS

Based on the evaluation of the design, testing, and analysis results, several conclusions can be drawn as follows: The specifications for the Smart Health device have been successfully designed, with a base dimension of 30x30 cm, a height of 200 cm, and using wood and aluminum alloy as materials. From the sensor accuracy tests on the Smart Health device, different accuracy values were obtained for each sensor, and the results can be calculated. Integration testing of the device with the Blynk application shows that the Smart Health Monitoring can be integrated well. All sensor readings were successfully transmitted and displayed accurately on the Blynk application, and the control of the MPX700DP sensor operates smoothly..

VI. LIMITATIONS AND FUTURE WORK

Based on the results of this Final Project, several recommendations for further development can be proposed as follows: including a respiration sensor to measure the respiration rate on the device, refining the device design to make it more portable and easier to move, implementing automatic blood pressure measurement using a motor pump, replacing the conventional pump cuff to improve efficiency, and updating the application to allow more users to monitor and control the device.

REFERENCES

[1] RB. Asyim and Yulianto, “Perilaku Konsumsi Obat Tradisional dalam Upaya Menjaga Kesehatan Masyarakat Bangsawan Sumenep,” *J. Keperawatan*, vol. Vol. 15, no. No. 2, p. 2, 2022, [Online]. Available: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiHv5LumJf9AhUE23MBHfP5B08QFnoECAsQAQ&url=https%3A%2F%2Fjournal.unhas.ac.id%2Findex.php%2Fjnik%2Farticle%2Fview%2F4281%2F2691&usg=AOvVaw18c01-ZoiAK3BpaKPz6BAw

[2] J. Birokrasi, K. dan Pelayanan Publik, A. Yulianti, B. S. Utoyo, and D. Brima Atika, “Kinerja Program Nusantara Sehat di Daerah Terpencil Perbatasan dan Kepulauan,” 2022.

[3] F. K. dan D. D. dan A. M. Vikry, “Rancang Bangun Alat Perekam Data Cuaca dengan Berbasis Internet Of Things : Design and Build Weather Data Recorder Based on Internet Of Things,” *J. Hidropilar*, vol. 6, no. 2, pp. 61–74, 2020, [Online]. Available: http://jurnal.sttalhidros.ac.id/index.php/hidropilar/article/view/215

[4] F. D. Silalahi, J. Dian, and N. D. Setiawan, “Implementasi Internet Of Things (Iot) Dalam Monitoring Suhu Dan Kelembaban Ruang Produksi Obat Non Steril Menggunakan Arduino Berbasis Web,” *J. JUPITER*, vol. 13, no. 2, pp. 62–68, 2021.

[5] S. S. Santoso and A. Andriyani, “Analisis Pelaksanaan Medical Check Up (MCU) pada Pegawai Rumah Sakit Islam Jakarta Pondok Kopi Tahun 2016,” *J. Kedokt. dan Kesehat.*, vol. 13, no. 2, p. 171, 2017, doi: 10.24853/jkk.13.2.171-182.

[6] M. A. Adrian, M. R. Widiarto, and R. S. Kusumadiarti, “Health Monitoring System dengan Indikator Suhu Tubuh, Detak Jantung dan Saturasi Oksigen Berbasis Internet of Things (IoT),” *J. Petik*, vol. 7, no. 2, pp. 108–118, 2021, doi: 10.31980/jpetik.v7i2.1230.

[7] M. A. Saputro, “Implementation of a Wireless Heart Rate and Body Temperature Monitoring System,” *Dev. Inf. Technol. Comput. Sci.*, vol. 1, no. 2, pp. 148–156, 2017.

[8] A. Apriansyah, A. Fauzi, and S. Faisal, “Penerapan Fuzzy Logic Untuk Menentukan Indeks Massa Tubuh (IMT) Berbasis Internet of Things (IoT),” *J. Media Inform. Budidarma*, vol. 7, no. 1, pp. 292–299, 2023, doi: 10.30865/mib.v7i1.5470.

[9] I. W. Suriana, I. G. A. Setiawan, and I. M. S. Graha, “Rancang Bangun Sistem Pengaman Kotak Dana Punia berbasis Mikrokontroler NodeMCU ESP32 dan Aplikasi Telegram,” *J. Ilm. Telsinas Elektro, Sipil dan Tek. Inf.*, vol. 4, no. 2, pp. 75–84, 2022, doi: 10.38043/telsinas.v4i2.3198.

[10] F. Y. Saputra, M. S. Al Amin, and . P., “Alat Pengukur Tinggi Badan, Berat Badan, Dan Suhu Badan Digital Menggunakan Sensor Ultrasonik, Load Cell, Dan Inframerah Mlx90614,” *J. Tekno*, vol. 19, no. 1, pp. 60–67, 2022, doi: 10.33557/jtekno.v19i1.1638.

[11] I. N. A. Yogiarditha, “Perancangan Alat ukur Tekanan Darah dan Jumlah Denyut Jantung Berbasis Arduino Uno,” *Jur. Tek. Elektro Univ. Mataram*, vol. 3, no. 1, pp. 1–12, 2019.

[12] B. Harianto, A. Hidayat, and F. N. Hulu, “ANALISIS PENGGUNAAN SENSOR MAX30100 PADA SISTEM PENDETEKSI DETAK JANTUNG BERBASIS IoT BLYNK,” *Semin. Nas. Teknol.*, vol. 2021, no. SemanTECH, pp. 238–245, 2021.

[13] H. A. Saputro, *Rancang Bangun Alat Pengukur Tinggi Badan Digital Dengan Sensor Ultrasonik HC-SR04 Berbasis Arduino UNO*. 2017.

[14] Y. D. S. Budoyo and A. D. Andriana, “Sistem Iot Timbangan Digital Menggunakan Sensor Load Cell Di Ud. Pangrukti Tani,” *Tek. Inform. - Univ. Komput. Indones.*, p. hal 8, 2018, [Online]. Available: https://elibrary.unikom.ac.id/id/eprint/1111/

[15] U. Khair and T. Sabrina, “Alat Pemberi Makan Kucing Otomatis Berbasis Arduino Uno Pada Pet Shop,” *Sebatik*, vol. 23, no. 1, pp. 9–14, 2019, doi: 10.46984/sebatik.v23i1.437.

[16] risky abadi, “Thecityfoundry Speaker : Pengertian , Fungsi , Jenis , Bagian , Cara Kerja,” pp. 1–11, 2023.

[17] K. J. Tengah, “Rancang Bangun Sistem Inventory Gudang Menggunakan Metode Waterfall (Studi Kasus Di CV . Aqualux Duspha Abadi,” pp. 74–82.