

# LAMPIRAN A

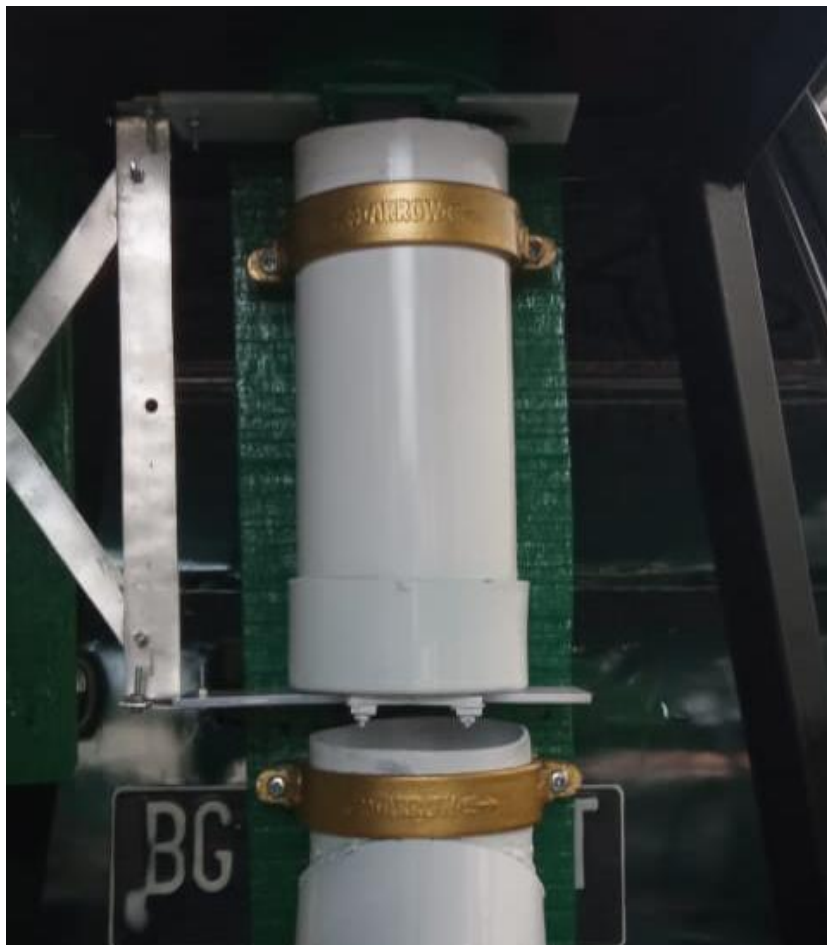
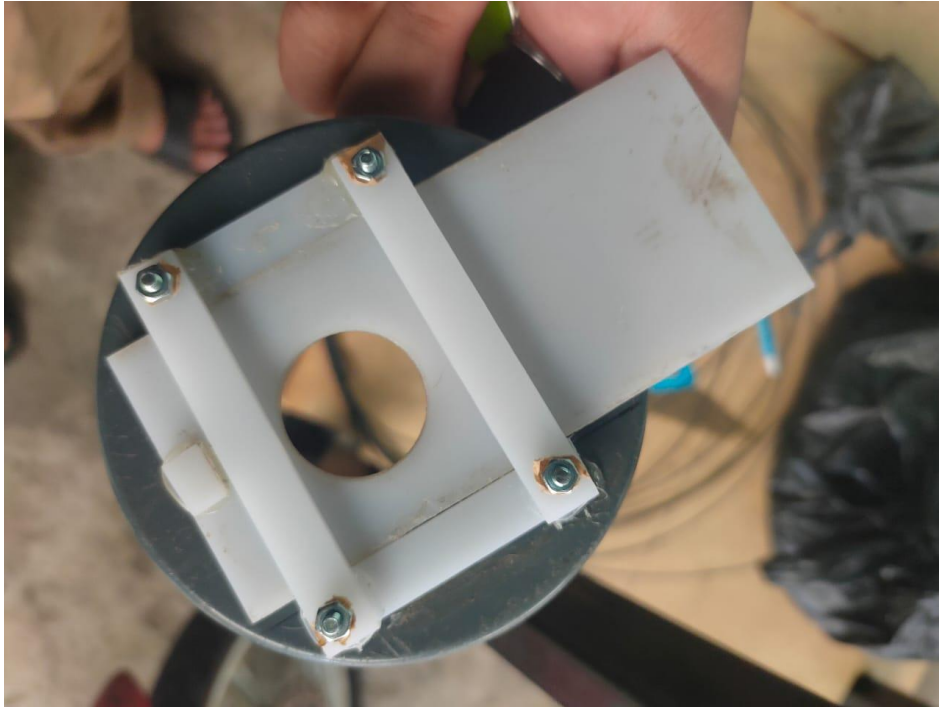
## 1. Bentuk Awal Mekanik Alat



## 2. Bentuk Akhir Mekanik Alat



### 3. Katup dan Tabung Penampung Pakan ½ Kg



#### 4. Feed Tank



#### 5. Pipa Penyebar pakan ke kolam



**6. Keadaan Alat Saat Melakukan pemberian pakan pada kolam**







### Blynk Notification

Alat Pakan Ikan Otomatis:  
feed\_alert  
Pakan pertama jam 08.00  
sudah diberikan

CLOSE

### Blynk Notification

Alat Pakan Ikan Otomatis:  
feed\_alert  
Pakan kedua jam 12.00  
sudah diberikan

CLOSE

SHOW DEVICE

### Blynk Notification

Alat Pakan Ikan Otomatis:  
feed\_alert  
Pakan ketiga jam 16.00  
sudah diberikan

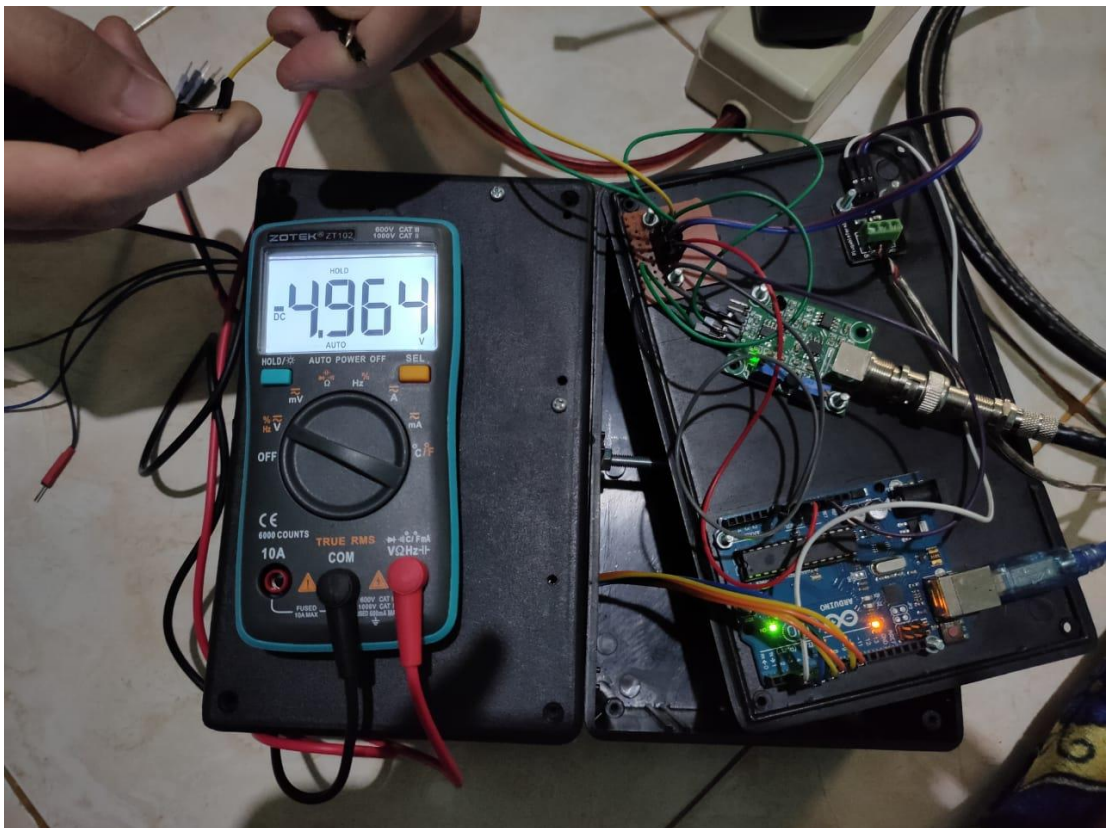
CLOSE

SHOW DEVICE



## 7. Pengambilan Data

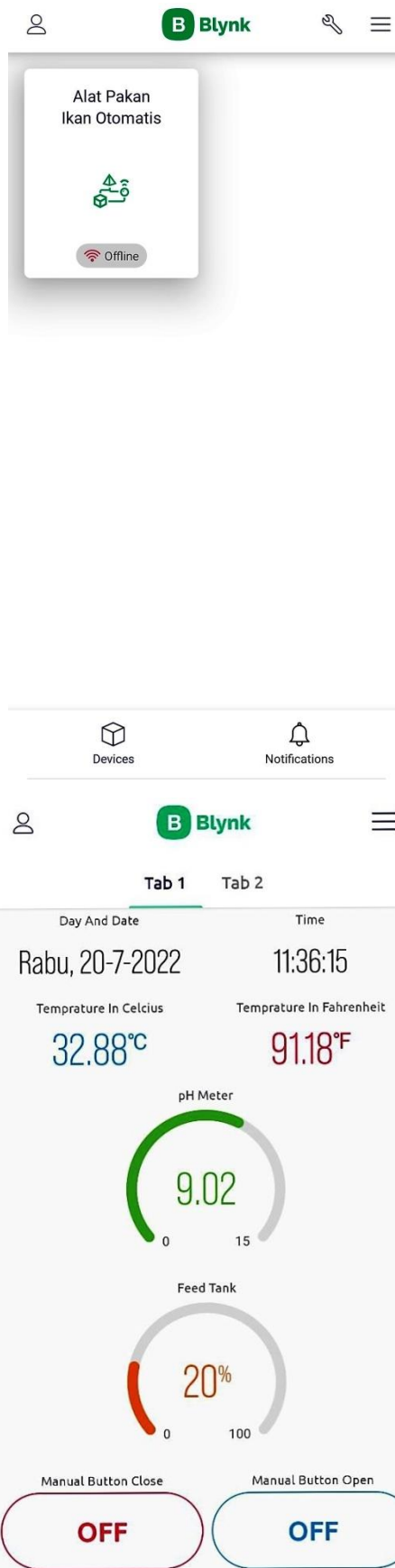








## 8. Tampilan Monitoring Di Aplikasi *Blynk*



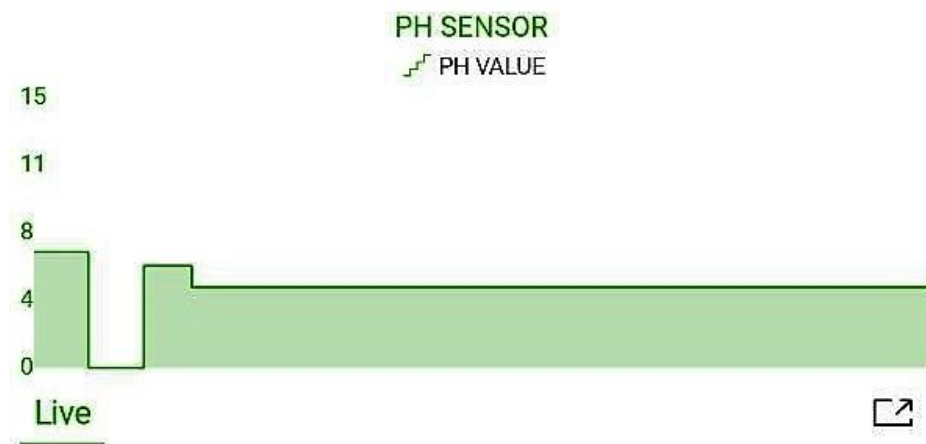
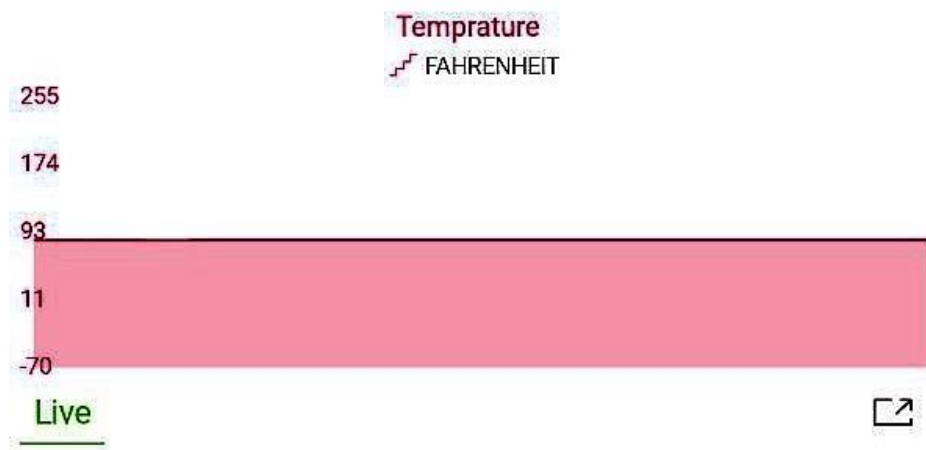
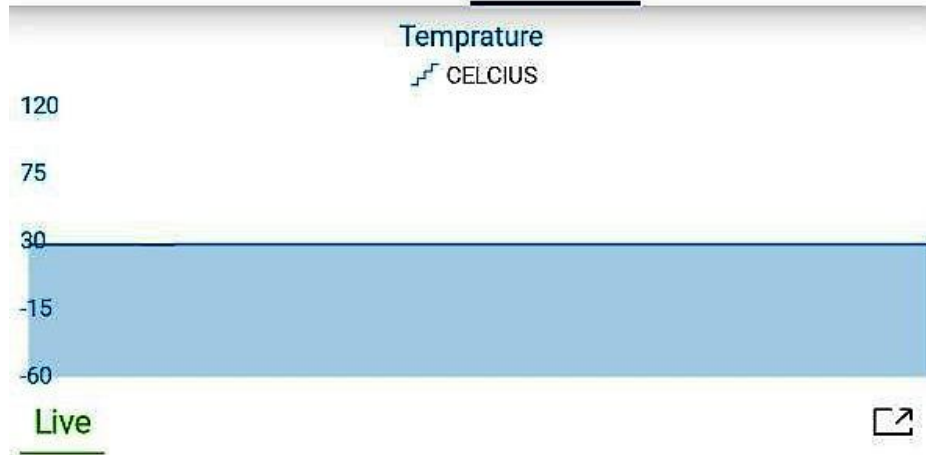


# Alat Pakan Ikan Otomatis



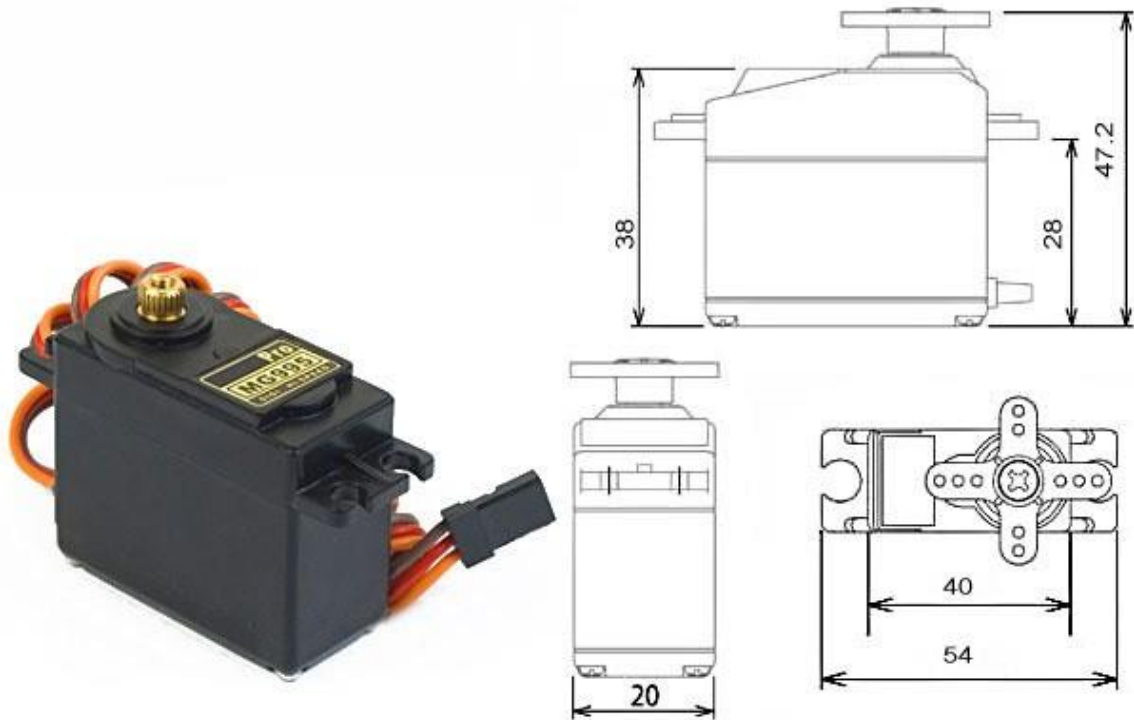
Tab 1

Tab 2



# **LAMPIRAN B**

# MG995 High Speed Metal Gear Dual Ball Bearing Servo



The unit comes complete with 30cm wire and 3 pin 'S' type female header connector that fits most receivers, including Futaba, JR, GWS, Cirrus, Blue Bird, Blue Arrow, Corona, Berg, Spektrum and Hitec.

This high-speed standard servo can rotate approximately 120 degrees (60 in each direction). You can use any servo code, hardware or library to control these servos, so it's great for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. The MG995 Metal Gear Servo also comes with a selection of arms and hardware to get you set up nice and fast!

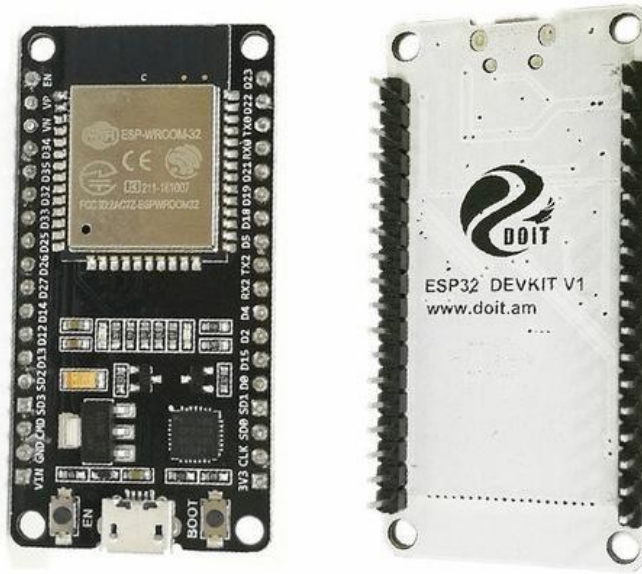
## Specifications

- Weight: 55 g
- Dimension: 40.7 x 19.7 x 42.9 mm approx.
- Stall torque: 8.5 kgf·cm (4.8 V), 10 kgf·cm (6 V)
- Operating speed: 0.2 s/60° (4.8 V), 0.16 s/60° (6 V)
- Operating voltage: 4.8 V a 7.2 V
- Dead band width: 5  $\mu$ s
- Stable and shock proof - double ball bearing design
- Temperature range: 0 °C – 55 °C




# DOIT Esp32 DevKit v1


The DOIT Esp32 DevKit v1 is one of the development board created by DOIT to evaluate the ESP-WROOM-32 module. It is based on the ESP32 microcontroller that boasts Wifi, Bluetooth, Ethernet and Low Power support all in a single chip.



## Pin Mapping

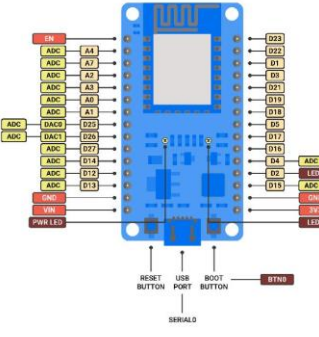


# DOIT ESP32 DevKit v1



PINOUT DIAGRAM

DO NOT USE D6 TO D11  
PWM IS ENABLED ON EVERY DIGITAL PIN  
ICU NOT SUPPORTED  
ADC ON PINS D4, D12, D13, D14, D15, D25, D26, D27  
CAN BE READ ONLY WITH WI-FI NOT STARTED



## Flash Layout

The internal flash of the ESP32 module is organized in a single flash area with pages of 4096 bytes each. The flash starts at address 0x00000, but many areas are reserved for Esp32 IDF SDK and Zerynth VM. There exist two different layouts based on the presence of BLE support.

In particular, for non-BLE VMs:

Start address	Size	Content
0x00009000	16Kb	Esp32 NVS area
0x0000D000	8Kb	Esp32 OTA data
0x0000F000	4Kb	Esp32 PHY data
0x00010000	1Mb	Zerynth VM
0x00110000	1Mb	Zerynth VM (FOTA)
0x00210000	512Kb	Zerynth Bytecode
0x00290000	512Kb	Zerynth Bytecode (FOTA)
0x00310000	512Kb	Free for user storage
0x00390000	448Kb	Reserved

For BLE VMs:

Start address	Size	Content
0x00009000	16Kb	Esp32 NVS area
0x0000D000	8Kb	Esp32 OTA data
0x0000F000	4Kb	Esp32 PHY data
0x00010000	1216Kb	Zerynth VM
0x00140000	1216Kb	Zerynth VM (FOTA)
0x00270000	320Kb	Zerynth Bytecode
0x002C0000	320Kb	Zerynth Bytecode (FOTA)
0x00310000	512Kb	Free for user storage
0x00390000	448Kb	Reserved

## Device Summary

- Microcontroller: Tensilica 32-bit Single-/Dual-core CPU Xtensa LX6
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 25
- Analog Input Pins (ADC): 6
- Analog Outputs Pins (DAC): 2
- UARTs: 3
- SPIs: 2
- I2Cs: 3
- Flash Memory: 4 MB
- SRAM: 520 KB
- Clock Speed: 240 Mhz
- Wi-Fi: IEEE 802.11 b/g/n/e/i:

- Integrated TR switch, balun, LNA, power amplifier and matching network
- WEP or WPA/WPA2 authentication, or open networks

#### Power

Power to the DOIT Esp32 DevKit v1 is supplied via the on-board USB Micro B connector or directly via the “VIN” pin. The power source is selected automatically.

The device can operate on an external supply of 6 to 20 volts. If using more than 12V, the voltage regulator may overheat and damage the device. The recommended range is 7 to 12 volts.

#### Connect, Register, Virtualize and Program

The DOIT Esp32 DevKit v1 comes with a serial-to-usb chip on board that allows programming and opening the UART of the ESP32 module. Drivers may be needed depending on your system (Mac or Windows) and can be download from the official Espressif documentation page. In Linux systems, the DevKit v1 should work out of the box.

## Arduino Uno



#### Overview

The Arduino Uno is a microcontroller board based on the ATmega328 ([datasheet](#)). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

Revision 2 of the Uno board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into [DFU mode](#).

Revision 3 of the board has the following new features:

1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that use the AVR, which operate with 5V and with the Arduino Due that operate with 3.3V. The second one is a not connected pin, that is reserved for future purposes.

Stronger RESET circuit.

Atmega 16U2 replace the 8U2.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the [index of Arduino boards](#).

#### Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

#### Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

**VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

**5V.** This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.

**3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

**GND.** Ground pins.

#### Memory

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the [EEPROM library](#)).

#### Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using [pinMode\(\)](#), [digitalWrite\(\)](#), and [digitalRead\(\)](#) functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

**Serial:** 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.

**External Interrupts:** 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the [attachInterrupt\(\)](#) function for details.

**PWM:** 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the [analogWrite\(\)](#) function.

**SPI:** 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the [SPI library](#).

**LED:** 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though it is possible to change the upper end of their range using the AREF pin and the [analogReference\(\)](#) function. Additionally, some pins have specialized functionality:

TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the [Wire library](#).

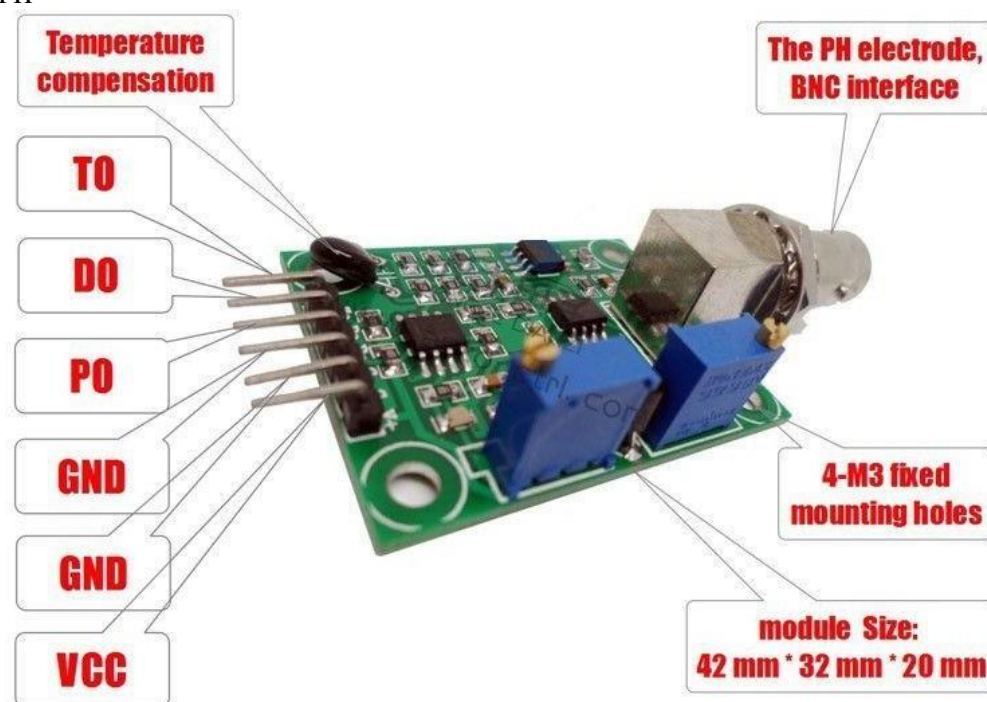
There are a couple of other pins on the board:

AREF. Reference voltage for the analog inputs. Used with [analogReference\(\)](#).

Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the [mapping between Arduino pins and ATmega328 ports](#). The mapping for the Atmega8, 168, and 328 is identical.

PH



PH Probe Sensor Pinout

- TO – Temperature output
- DO – 3.3V Output (from ph limit pot)
- PO – PH analog output ==> Arduino A0
- Gnd – Gnd for PH probe (can come from Arduino GND pin) ==> Arduino GND Gnd – Gnd for board (can also come from Arduino GND pin) ==> Arduino GND VCC – 5V DC (can come from Arduino 5V pin) ==> Arduino 5V pin
- POT 1 – Analog reading offset (Nearest to BNC connector) POT 2 – PH limit setting

PH probe module Offset and how to use it.

This board have the ability to supply a voltage output to the analogue board that will represent a PH value just like any other sensor that will connect to an analog pin. Ideally, we want a PH 0 represent 0v and a PH of 14 to represent 5V.

BUT there is a catch....., this board by default have PH 7 set to 0V (or near it, it differs from one PH probe to another, that is why we have to calibrate the probe as you will see later on), This

means that the voltage will go into the minuses when reading acidic PH values and that cannot be read by the analog Arduino port. The offset pot is used to change this so that a PH 7 will read the expected 2.5V to the Arduino analog pin, the analog pin can read voltages between 0V and 5V hence the 2.5V that is halfway between 0V and 5V as a PH 7 is halfway between PH 0 and PH 14, You will need to turn the offset potentiometer to get the right offset, The offset pot is the blue pot nearest to the BNC connector.

To set the offset is easy. First, you need to disconnect the probe from the circuit and short-circuit the inside of the BNC connector with the outside to simulate a neutral PH (PH7). I took a piece of wire, strip both sides, wrap the one side around the outside of the BNC connector and push the other side into the BNC hole. This short-circuit represents about a neutral PH reading of 7.



**MANTECH  
ELECTRONICS**

*International Components Distributor*  
A MOBICON COMPANY

TEL JHB : (011) 493-9307  
CAPE : (021) 535-3150  
KZN : (031) 309-7686

FAX : (011) 493-9319

sales@mantech.co.za

www.mantech.co.za



## I2C interface for LCD

Discription:

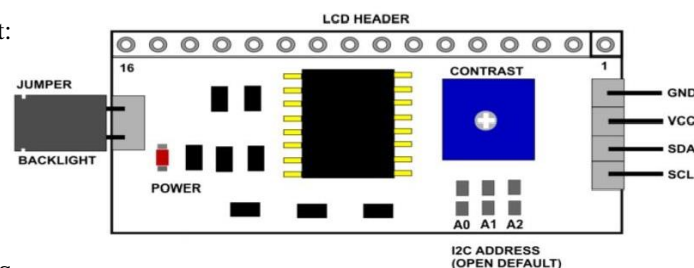
This LCD2004 is a great I2C interface for 2x16 and 4x20 LCD displays. With the limited pin resources, your project may be out of resources using normal LCD shield. With this I2C interface LCD module, you only need 2 lines (I2C) to display the information. If you already has I2C devices in your project, this LCD module actually cost no more resources at all. Fantastic for Arduino based projects.

Specification:

Compatible with 16x2 and 20x4 LCD's Default I2C Address = 0X27

Address selectable - Range 0x20 to 0x27

Board Layout:



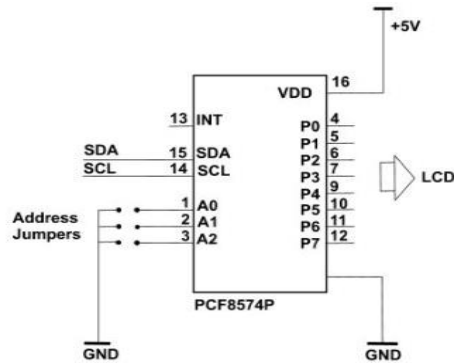
I2C Address Setup:

The LCD2004 board utilized the PCF8574 I/O expander. This nifty little chip provides eight bits of parallel I/O addressabl by a I2C bus address – 0x00 to 0x27. SainSmart tied all address leads to Vcc, so the LCD2004 board's I2C addressis permanently fixed at hex 27. This is rather limiting

since no additional LCD2004s can be added to the bus. Anyway, you simply address the board and write an eight bit value which is then presented on the output pins of the PCF8574, which, in this case, are connected to the HD44780 based LCD screen.

INPUTS			I2C SLAVE ADDRESS
A2	A1	A0	
L	L	L	0x20
L	L	H	0x21
L	H	L	0x22
L	H	H	0x23
H	L	L	0x24
H	L	H	0x25
H	H	L	0x26
H	H	H	0x27

H = Open Jumper L = Close Jumper



Tech Support: [services@elecfreaks.com](mailto:services@elecfreaks.com)

### Ultrasonic Ranging Module HC - SR04

Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:

- (1) Using IO trigger for at least 10us high level signal,
- (2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
- (3) IF the signal back, through high level, time of high output IO duration is the time from sending ultrasonic to returning.

Test distance = (high level time × velocity of sound (340M/S) / 2,

Wire connecting direct as following:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

Electric Parameter

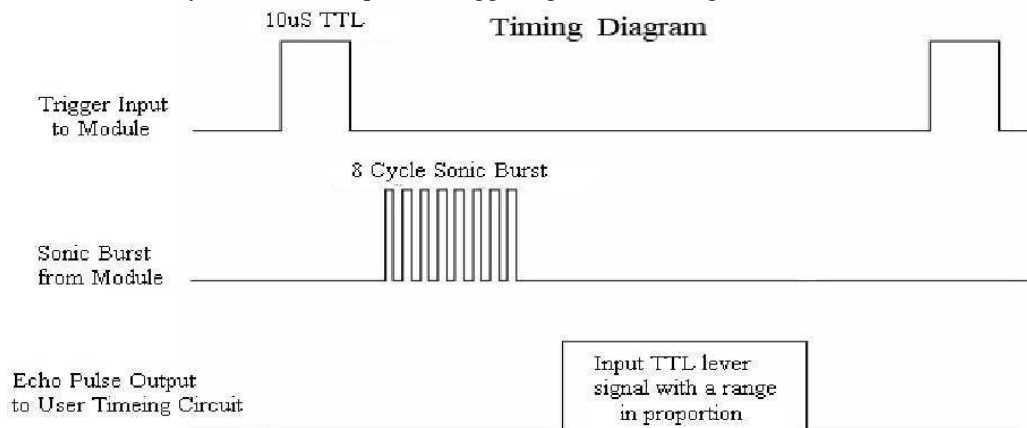
<b>Working Voltage</b>	<b>DC 5 V</b>
<b>Working Current</b>	<b>15mA</b>
<b>Working Frequency</b>	<b>40Hz</b>
<b>Max Range</b>	<b>4m</b>
<b>Min Range</b>	<b>2cm</b>
<b>Measuring Angle</b>	<b>15 degree</b>

<b>Trigger Input Signal</b>	<b>10uS TTL pulse</b>
<b>Echo Output Signal</b>	<b>Input TTL lever signal and the range in proportion</b>
<b>Dimension</b>	<b>45*20*15mm</b>



### Timing diagram

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion. You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula:  $\mu\text{s} / 58 = \text{centimeters}$  or  $\mu\text{s} / 148 = \text{inch}$ ; or:  $\text{range} = \text{high level time} * \text{velocity} (340\text{M/S}) / 2$ ; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to echo signal.



### Attention:

- The module is not suggested to connect directly to electric, if connected electric, the GND terminal should be connected the module first, otherwise, it will affect the normal work of the module.
- When tested objects, the range of area is not less than 0.5 square meters and the plane requests as smooth as possible, otherwise, it will affect the results of measuring.





## DS18B20 Waterproof Temperature Sensor Cable



### Product Description

This Maxim-made item is a digital thermo probe or sensor that employs DALLAS DS18B20. Its unique 1-wire interface makes it easy to communicate with devices. It can convert temperature to a 12-bit digital word in 750ms (max). Besides, it can measure temperatures from  $-55^{\circ}\text{C}$  to  $+125^{\circ}\text{C}$  ( $-67\text{F}$  to  $+257\text{F}$ ). In addition, this thermo probe doesn't require any external power supply since it draws power from the data line. Last but not least, like other common thermo probes, its stainless steel probe head makes it suitable for any wet or harsh environment.

The datasheet of this DS18B20 Sensor can be found from:

<https://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Temp/DS18B20.pdf>

### Feature:

Power supply range:	3.0V to 5.5V
Operating temperature range:	$-55^{\circ}\text{C}$ to $+125^{\circ}\text{C}$ ( $-67\text{F}$ to $+257\text{F}$ )
Storage temperature range:	$-55^{\circ}\text{C}$ to $+125^{\circ}\text{C}$ ( $-67\text{F}$ to $+257\text{F}$ )
Accuracy over the range of $-10^{\circ}\text{C}$ to $+85^{\circ}\text{C}$ :	$\pm 0.5^{\circ}\text{C}$
3-pin 2510 Female Header Housing	
Waterproof Stainless steel sheath	
Stainless steel sheath	
Size of Sheath:	6*50mm
Connector:	RJ11/RJ12, 3P-2510, USB.
Pin Definition:	RED: VCC Yellow: DATA Black: GND
Cable length:	1meter, 2m, 3m, 4m are available upon request

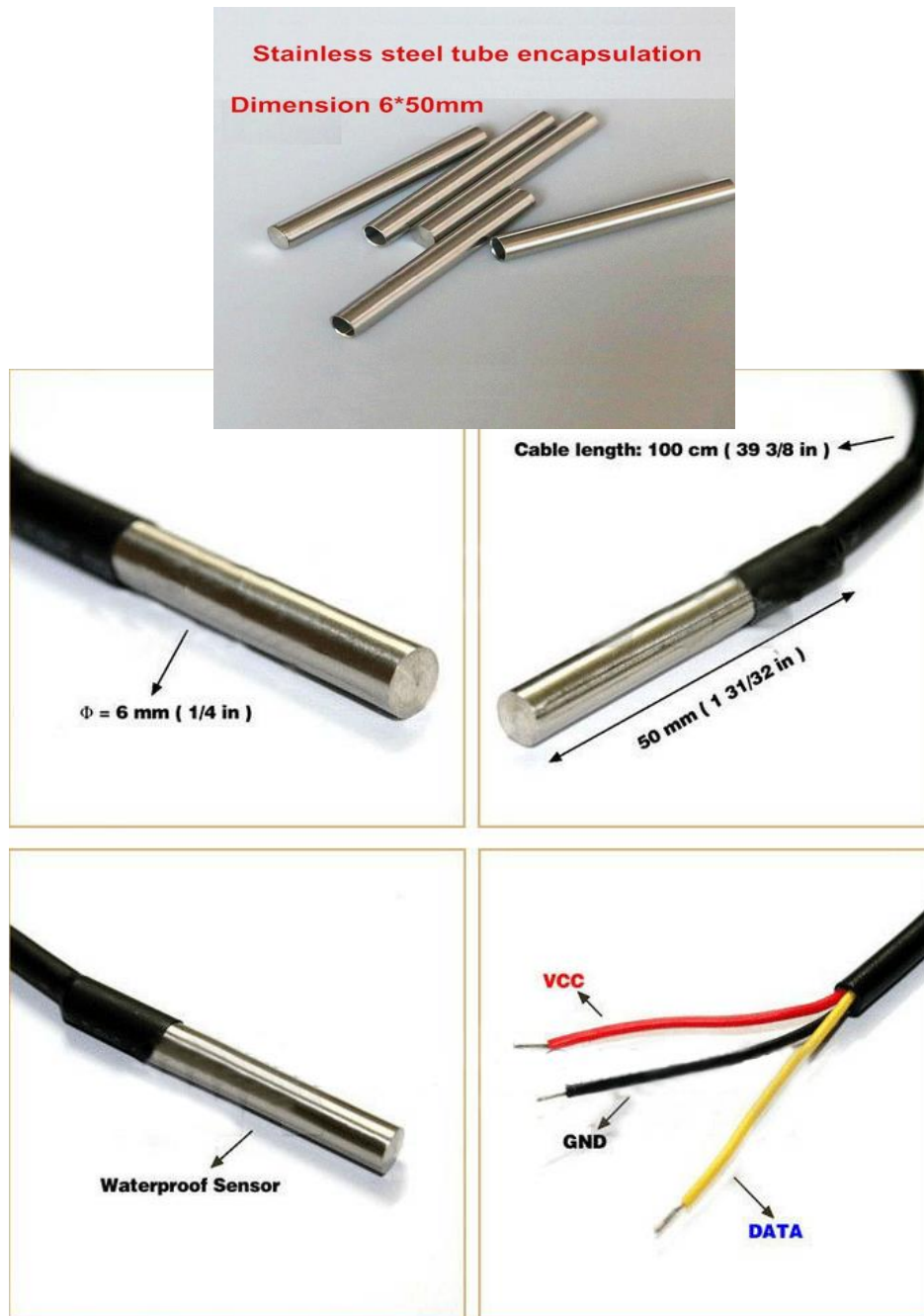
### Application:

The DS18B20 Digital Temperature Probe provides 9 to 12 bit

(configurable) temperature readings which indicate the temperature of the device. Information is sent to/from the DS18B20 over a 1-Wire interface, so that only one wire (and ground) needs to be connected from a central microprocessor to a DS18B20. Power for reading, writing, and performing temperature conversions can be derived from the data line itself with no need for an external power source.

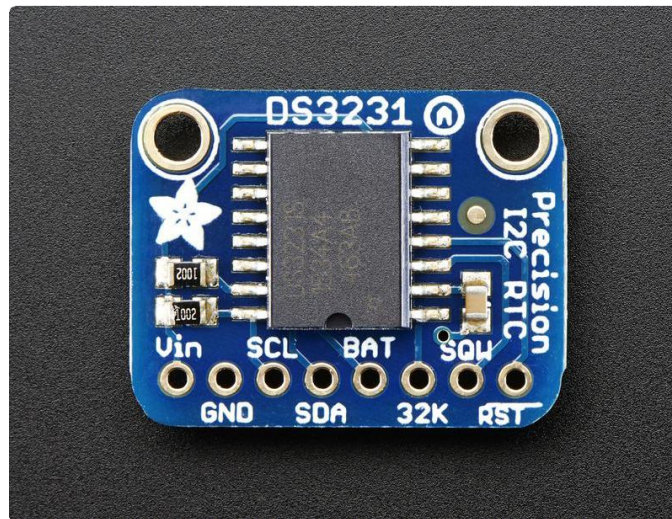
Because each DS18B20 contains a unique silicon serial number, multiple DS18B20s can exist on the same 1Wire bus. This allows for placing temperature sensors in many different places. Applications where this feature is useful include HVAC environmental controls, sensing temperatures inside buildings, equipment or machinery, and process *Monitoring* and control.

Details :





Adafruit DS3231 Precision RTC Breakout

**Power Pins:**

**Vin** - this is the power pin. Since the RTC can be powered from 2.3V to 5.5V power, you do not need a regulator or level shifter for 3.3V or 5V logic/power. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V micro like Arduino, use 5V

**GND** - common ground for power and logic

**I2C Logic pins:**

- **SCL** - I2C clock pin, connect to your microcontrollers I2C clock line. This pin has a 10K pullup resistor to Vin
- **SDA** - I2C data pin, connect to your microcontrollers I2C data line. This pin has a 10K pullup resistor to Vin
- **STEMMA QT** (<https://adafru.it/Ft4>) - On the STEMMA QT version only! These connectors allow you to connect to development boards with STEMMA QT connectors, or to other things, with [various associated accessories](https://adafru.it/Ft6) (<https://adafru.it/Ft6>) .

**Other Pins:**

**BAT** - this is the same connection as the positive pad of the battery. You can use this if you want to power something else from the coin cell, or provide battery backup from a different separate battery. VBat can be between 2.3V and 5.5V and the DS3231 will switch over when main Vin power is lost

**32K** - 32KHz oscillator output. Open drain, you need to attach a pullup to read this signal from a microcontroller pin

**SQW** - optional square wave or interrupt output. Open drain, you need to attach a pullup to read this signal from a microcontroller pin

**RST** - This one is a little different than most RST pins, rather than being just an input, it is designed to be used to reset an external device or indicate when main power is lost. Open drain, but has an internal 50K pullup. The pullup keeps this pin voltage high as long as Vin is present. When Vin drops and the chip switches to battery backup, the pin goes low.

# LAMPIRAN C

## 1. Pemograman Di Arduino Uno

```
#include <SoftwareSerial.h>
#include <OneWire.h>
#include <DallasTemperature.h>

#define ONE_WIRE_BUS 2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

//----- Komunikasi Serial
SoftwareSerial SuhuC(3, 4);// RX, TX
SoftwareSerial SuhuF(5, 6);// RX, TX
SoftwareSerial Phsensor(7, 8);// RX, TX

//----- pH Sensor
const int ph_Pin = A0;
float Po = 0;
float PH_step;
int nilai_analog_PH;
double TeganganPh;

//kalibrasi
float PH4 = 2.3;
float PH7 = 2.1;

void setup(void){
  Serial.begin(9600);
  pinMode (ph_Pin, INPUT);
  sensors.begin();
  SuhuC.begin(9600);
  SuhuF.begin(9600);
  Phsensor.begin(9600);
}

void loop(void){
  //----- DS18B20 Sensor
  sensors.requestTemperatures();
  //Celcius
  Serial.print("Celsius temperature: ");
  Serial.print(sensors.getTempCByIndex(0));
  SuhuC.println(sensors.getTempCByIndex(0));

  //Fahrenheit
  Serial.print(" - Fahrenheit temperature: ");
  Serial.println(sensors.getTempFByIndex(0));
  SuhuF.println(sensors.getTempFByIndex(0));
}
```

```

int tempC=sensors.getTempCByIndex(0);
int tempF=sensors.getTempFByIndex(0);

//----- pH Sensor
nilai_analog_PH = analogRead(ph_Pin);
//Serial.print("Nilai ADC Ph: ");
//Serial.println(nilai_analog_PH);
TeganganPh = 5 / 1023.0 * nilai_analog_PH;
Serial.print("TeganganPh: ");
Serial.println(TeganganPh,3);

PH_step = (PH4 - PH7) / 3;
Po = 7.00 + ((PH7 - TeganganPh) / PH_step);
Serial.print("Nilai PH Cairan: ");
Serial.println(Po, 2);
Phsensor.println(Po, 2);
delay(1000);
}

```

## 2. Pemograman Di NodeMCU ESP32

```

#include "RTClib.h"
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <BlynkSimpleEsp32.h>
#include "SoftwareSerial.h"

//
SoftwareSerial SuhuC(35, 34);// RX, TX
SoftwareSerial SuhuF(33, 32);// RX, TX
SoftwareSerial Phsensor(26 , 25);// RX, TX

#define BLYNK_TEMPLATE_ID "TMPLLeLy4v2UU"
#define BLYNK_DEVICE_NAME "Alat Pakan Ikan Otomatis"
#define BLYNK_AUTH_TOKEN "uu8SPNb5Ukh0WdtQRQrEHaG6rNiariNs"
#define BLYNK_PRINT Serial

char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "realme 5 Pro";
char pass[] = "00001111";

```

```

//Ultra
const int trigPin = 13;
const int echoPin = 12;
long tankDepth=45;

//RTC
RTC_DS3231 rtc;
char dataHari[7][12] = {"Minggu", "Senin", "Selasa", "Rabu",
"Kamis", "Jumat", "Sabtu"};
String hari;
int tanggal, bulan, tahun, jam, menit, detik;
float suhu;

//LCD
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

//servo
#include <Servo.h>
Servo mekanik;

//Manual Tombol
BLYNK_WRITE(V6){
    mekanik.write(20);
}
BLYNK_WRITE(V7){
    mekanik.write(150);
}

////////////////////////////////////
////////////////////////////////////
void setup () {
    Serial.begin(9600);

```

```
SuhuC.begin(9600);
SuhuF.begin(9600);
Phsensor.begin(9600);
//Ultra
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input

//LCD
  lcd.begin();

//servo
  mekanik.attach(27);
  mekanik.write(20);

//RTC
  if (! rtc.begin()) {
    Serial.println("RTC Tidak Ditemukan");
    Serial.flush();
    abort();
  }
//Atur Waktu
//rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
//rtc.adjust(DateTime(2022, 6, 12, 14, 40, 0));

//ESP
  lcd.clear();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, pass);
  int wifi_ctr = 0;
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi connected");
```



```

    Blynk.begin(auth, ssid, pass, "sgp1.blynk.cloud", 80);
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
void loop () {
//ultrasonic
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
long t = pulseIn(echoPin, HIGH);
    long cm = t / 29 / 2;
    double level= tankDepth-cm;
    if(level>0){
        long percentage=((level/tankDepth))*100;
        Blynk.virtualWrite(V2,percentage);
        Serial.println(percentage);
    }

//Blynk
    kirimdatablynk();

//RTC
    DateTime now = rtc.now();
    hari    = dataHari[now.dayOfTheWeek()];
    tanggal = now.day(), DEC;
    bulan   = now.month(), DEC;
    tahun   = now.year(), DEC;
    jam     = now.hour(), DEC;
    menit   = now.minute(), DEC;
    detik   = now.second(), DEC;
    suhu    = rtc.getTemperature();

```

```

    Serial.println(String() + hari + "," + tanggal + "-" + bulan
+ "-" + tahun);
    Serial.println(String() + jam + ":" + menit + ":" + detik);

//LCD
    lcd.setCursor(0, 0);
    lcd.print (String() + hari + "," + tanggal + "-" + bulan + "-"
+ "-" + tahun);
    lcd.print(" ");
    lcd.setCursor(4, 1);
    lcd.print (String() + jam + ":" + menit + ":" + detik);
    lcd.print(" ");

//kondisi
    if(jam == 8 & menit == 00 & detik == 0){
        lcd.clear();
        lcd.setCursor(2, 0);
        lcd.print("Beri Pakan 1");//untuk tampilan saat waktu makan
        //Blynk.email("pakanikaniot@gmail.com","Alert","FEED 1 HAS
BEEN GIVEN");
        Blynk.logEvent("feed_alert","Pakan pertama jam 08.00 sudah
diberikan");
        delay(200);
        kasih_pakan(3);
    }

    if(jam == 12 & menit == 00 & detik == 0){
        lcd.clear();
        lcd.setCursor(2, 0);
        lcd.print("Beri Pakan 2");//untuk tampilan saat waktu makan
        //Blynk.email("pakanikaniot@gmail.com","Alert","FEED 2 HAS
BEEN GIVEN");
        Blynk.logEvent("feed_alert","Pakan kedua jam 12.00 sudah
diberikan");
    }

```

```

delay(200);
kasih_pakan(3);
}

if(jam == 16 & menit == 00 & detik == 0){
lcd.clear();
lcd.setCursor(2, 0);
lcd.print("Beri Pakan 3");//untuk tampilan saat waktu makan
//Blynk.email("pakanikaniot@gmail.com","Alert","FEED 3 HAS
BEEN GIVEN");
  Blynk.logEvent("feed_alert","Pakan ketiga jam 16.00 sudah
diberikan");
  delay(200);
  kasih_pakan(3);
}

//suhu dan PH
//----- Data temprature Celcius
String data_suhuC = "";
while (SuhuC.available())
{
  data_suhuC += char(SuhuC.read());
}
//----- Data temprature Fahrenheit
String data_suhuF = "";
while (SuhuF.available())
{
  data_suhuF += char(SuhuF.read());
}
//----- Data Ph sensor
String data_Phsensor = "";
while (Phsensor.available())
{
  data_Phsensor += char(Phsensor.read());
}

```

```

//----- Agar tidak ada sepaasi di serial
monitor
data_suhuC.trim();
data_suhuF.trim();
data_Phsensor.trim();
//----- Komunikasi Serial Ke Blynk
Serial.println("Celsius temperature: " + data_suhuC );
Serial.println("Fahrenheit temperature: " + data_suhuF );
Serial.println("Nilai pH cairan: " + data_Phsensor );
Blynk.virtualWrite(V3, data_suhuC);
Blynk.virtualWrite(V4, data_suhuF);
Blynk.virtualWrite(V5, data_Phsensor);
delay(100);

}

////////////////////////////////////
////////////////////////////////////
//servo
void kasih_pakan(int jumlah){
  for(int i = 1; i <= jumlah; i++){
    mekanik.write(150);
    delay(13000);
    mekanik.write(20);
    delay(19000);
  }
}
void kirimdatablynk()
{
  Blynk.virtualWrite(V0, (String() + hari + ", " + tanggal + "-"
+ bulan + "-" + tahun));
  Blynk.virtualWrite(V1, (String() + jam + ":" + menit + ":" +
detik));
  Blynk.run();
}

```

# **LAMPIRAN D**