Lampiran 01

***Source code* untuk TEST_MEGA 2560**

```
#include <MemoryFree.h>
#include <EEPROM.h>

#define FIRST_PIN 0   // первый вывод
#define LAST_PIN 53   // последний вывод
#define PIN_LED  13   // вывод светодиода

void PinTest1(byte pin)
{
 if(pin < 10) Serial.print("PIN:  ");
 else Serial.print("PIN: ");
 Serial.print(pin);
 pinMode(pin, OUTPUT);
 digitalWrite(pin, 0);
 Serial.print("    LOW: ");
 if(!digitalRead(pin)) Serial.print("OK  ");
 else Serial.print("FAIL");
 digitalWrite(pin, 1);
 Serial.print("  HIGH: ");
 if(digitalRead(pin)) Serial.print("OK  ");
 else Serial.print("FAIL");
 pinMode(pin, INPUT);
 Serial.print("  PULL UP: ");
 if(digitalRead(pin)) Serial.print("OK  ");
 else Serial.print("FAIL");
 digitalWrite(pin, 0);
```

```
}

void PinTest2(byte pin)
{
  Serial.print("     ");
  pinMode(pin, OUTPUT);
  digitalWrite(pin, 1);
  delay(5);
  if(!digitalRead(pin))Serial.println("SHORT");
  else Serial.println("OK");
  pinMode(pin, INPUT);
  digitalWrite(pin, 0);

}

void EEPROMTest() {

      }

void displayHelp() {
  Serial.println(F("\nArduino hardware test"));
  Serial.println(F("\ta = Blink test"));
  Serial.println(F("\tb = EEPROM test"));
  Serial.println(F("\tc = Pins test"));
  Serial.println(F("\t? = Help page"));
  Serial.println();
}

void setup() {
```

```
  Serial.begin(115200);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }
  displayHelp();
}

void loop() {
  if (Serial.available() > 0) {
    switch (Serial.read()) {
      case 'a': // Тест светодиодом
        Serial.print(F("\nBlink test:"));
        Serial.print(F("\n\tStart blinking - please check the board led\n\t"));
        for(byte i = 1; i <= 10; i++) {
          digitalWrite(PIN_LED, HIGH);
          delay(1000);
          Serial.print(".");
          digitalWrite(PIN_LED, LOW);
          delay(1000);
          Serial.print(F("."));
        }
        Serial.print("\n\tStop blinking\n");
        Serial.println();
        displayHelp();
      break;
      case 'b':
        Serial.print(F("\nEEPROM test:\n"));
        Serial.print("\tSRAM free size: ");
        Serial.print(freeMemory());
```

```
    Serial.print(F(" bytes\n"));
    Serial.print("\tEEPROM size: ");
    Serial.print(EEPROM.length());
    Serial.print(F(" bytes\n"));
    Serial.println();
    displayHelp();
  break;
  case 'c': // Тест на короткое замыкание выводов
    Serial.print(F("\nTest of short circuit on GND or VCC and between pins:\n"));
    for(byte i = FIRST_PIN; i <= LAST_PIN; i++) {
      for(byte j = FIRST_PIN; j <= LAST_PIN; j++) {
        pinMode(j, INPUT);
        digitalWrite(j, 0);
      }
      Serial.print(F("\t"));
      PinTest1(i);
      for(byte j = FIRST_PIN; j <= LAST_PIN; j++) {
        pinMode(j, OUTPUT);
        digitalWrite(j, 0);
      }
      PinTest2(i);
    }
    for(byte j = FIRST_PIN; j <= LAST_PIN; j++) {
      pinMode(j, INPUT);
      digitalWrite(j, 0);
    }
    displayHelp();
  break;
  case '?':
```

```
        displayHelp();
      break;
      default:
        Serial.println();
      break;
    }
  }
}
```

Lampiran 02

***Source code* TEST-ESP8266**

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>

// впишите сюда данные, соответствующие вашей сети:
const char* ssid = "CPH1801";
const char* password = "1sampai8";

ESP8266WebServer server(80);
MDNSResponder mdns;

String webPage = "";

int led_pin = 13;

void setup(void){

 // подготовка:
 pinMode(led_pin, OUTPUT);
 digitalWrite(led_pin, LOW);
 Serial.begin(115200);
 while (!Serial) {
  ; // wait for serial port to connect. Needed for native USB port only
 }
```

```
// информация о контроллере
Serial.println("");
Serial.println("ESP8266 board info:");
Serial.print("\tChip ID: ");
Serial.println(ESP.getFlashChipId());
Serial.print("\tCore Version: ");
Serial.println(ESP.getCoreVersion());
Serial.print("\tChip Real Size: ");
Serial.println(ESP.getFlashChipRealSize());
Serial.print("\tChip Flash Size: ");
Serial.println(ESP.getFlashChipSize());
Serial.print("\tChip Flash Speed: ");
Serial.println(ESP.getFlashChipSpeed());
Serial.print("\tChip Speed: ");
Serial.println(ESP.getCpuFreqMHz());
Serial.print("\tChip Mode: ");
Serial.println(ESP.getFlashChipMode());
Serial.print("\tSketch Size: ");
Serial.println(ESP.getSketchSize());
Serial.print("\tSketch Free Space: ");
Serial.println(ESP.getFreeSketchSpace());

// тело веб-страницы
webPage += "<h1>ESP8266 Web Server</h1>";
webPage += "<p>Chip ID: ";
webPage += ESP.getFlashChipId();
webPage += "</p>";
webPage += "<p>Core Version: ";
webPage += ESP.getCoreVersion();
```

```
  webPage += "</p>";
  webPage += "<p>Chip Real Size: ";
  webPage += ESP.getFlashChipRealSize()/1024;
  webPage += " Kbit</p>";
  webPage += "<p>Chip Size: ";
  webPage += ESP.getFlashChipSize()/1024;
  webPage += " Kbit</p>";
  webPage += "<p>Chip Flash Speed: ";
  webPage += ESP.getFlashChipSpeed()/1000000;
  webPage += " MHz</p>";
  webPage += "<p>Chip Work Speed: ";
  webPage += ESP.getCpuFreqMHz();
  webPage += " MHz</p>";
  webPage += "<p>Chip Mode: ";
  webPage += ESP.getFlashChipMode();
  webPage += "</p>";
  webPage += "<p>LED state <a
href=\"LedON\"><button>ON</button></a> <a
href=\"LedOFF\"><button>OFF</button></a></p>";


  // подключение к WiFi
  WiFi.begin(ssid, password);
  Serial.println("");


  // ожидание соединения:
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("");
```

```
Serial.print("Connected to ");  //  "Подключились к "
Serial.println(ssid);
Serial.print("IP address: ");  //  "IP-адрес: "
Serial.println(WiFi.localIP());

//  Проверка запуска MDNS
if (mdns.begin("esp8266", WiFi.localIP())) {
  Serial.println("MDNS responder started");
}

server.on("/", []{
  server.send(200, "text/html", webPage);
});

server.on("/LedON", []{
  server.send(200, "text/html", webPage);
  digitalWrite(led_pin, HIGH);
  Serial.println("[ON]");
  delay(1000);
});

server.on("/LedOFF", []{
  server.send(200, "text/html", webPage);
  digitalWrite(led_pin, LOW);
  Serial.println("[OFF]");
  delay(1000);
});

server.begin();
```

```
  Serial.println("HTTP server started");

}

void loop(void){
  server.handleClient();
}
```

Lampiran 03

***Source code* TEST_MEGA-ESP**

```
#include <MemoryFree.h>
#include <EEPROM.h>

#define PIN_LED 13    // вывод светодиода
String inString;

// Настройка
void setup() {
  // Инициализация портов и выходов
  Serial.begin(115200);
  Serial3.begin(115200);
  pinMode(PIN_LED, OUTPUT);
  digitalWrite(PIN_LED, LOW);
}

// Выполнение
void loop() {
}

// Проверка события на порту Serial3
void serialEvent3() {
  while (Serial3.available()) {
    // Чтение данных из порта Serial3
    char inChar = Serial3.read();
    // Вывод прочитанных данных в порт Serial
    Serial.write(inChar);
```

```
    // Поиск команды в полученных данных (команда должна быть в квадратных скобках)
    inString += inChar;
    if (inChar == ']') {
      if (inString.indexOf("[ON]")>0) {
        digitalWrite(PIN_LED, HIGH);
      }
      else if (inString.indexOf("[OFF]")>0) {
        digitalWrite(PIN_LED, LOW);
      }
      else
      {
        Serial.println("Wrong command");
      }
      inString = "";
    }
  }
}
```

Lampiran 04

***Source Code test* Sensor DHT11**

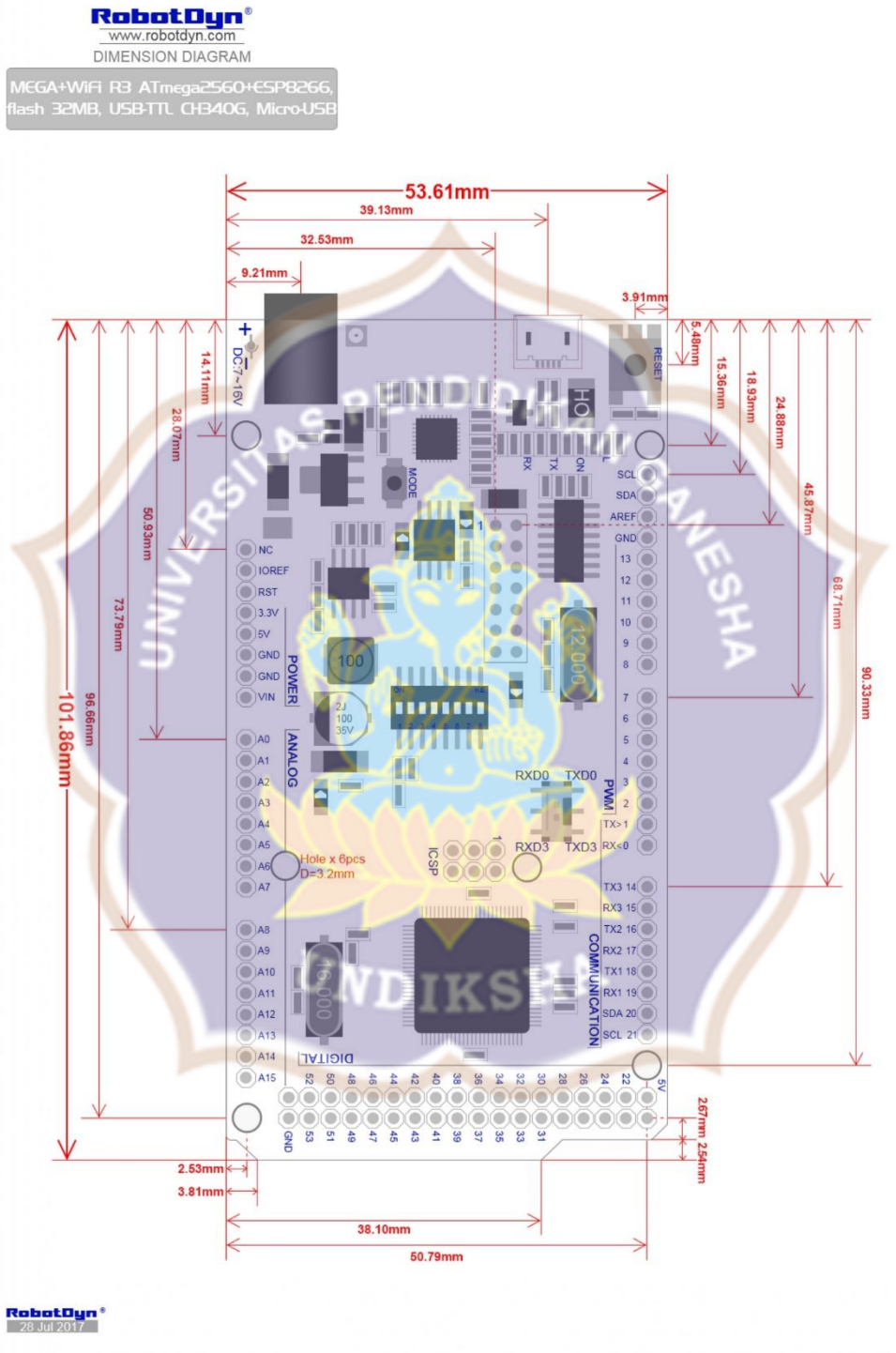
```
#define BLYNK_PRINT Serial

#include <ESP8266_Lib.h>
#include <BlynkSimpleShieldEsp8266.h>
#include <DHT.h>
#include <Adafruit_Sensor.h>

char auth[] = "CPLYENOj8Q1nj-ogp6wIfJEyCObPrYf1";

char ssid[] = "Fisika";
char pass[] = "Fisika123";
#define EspSerial Serial3
#define ESP8266_BAUD 115200
ESP8266 wifi(&EspSerial);

#define DHTPIN A0          // D3

#define DHTTYPE DHT11     // DHT 11
//#define DHTTYPE DHT22   // DHT 22, AM2302, AM2321
//#define DHTTYPE DHT21   // DHT 21, AM2301

DHT dht(DHTPIN, DHTTYPE);
BlynkTimer timer;

// This function sends Arduino's up time every second to Virtual Pin (A0).
// In the app, Widget's reading frequency should be set to PUSH. This means
// that you define how often to send data to Blynk App.
void sendSensor()
{
```

```
  float h = dht.readHumidity();
  float t = dht.readTemperature(); // or dht.readTemperature(true) for Fahrenheit

  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  // You can send any value at any time.
  // Please don't send more that 10 values per second.
  Blynk.virtualWrite(V7, t);
  Blynk.virtualWrite(V8, h);
}

void setup()
{
  Serial.begin(9600);
  Serial.begin(115200);
  Serial3.begin(115200);

  delay(10);
  EspSerial.begin(ESP8266_BAUD);
  delay(10);

 Blynk.begin(auth, wifi, ssid, pass, "blynk-cloud.com", 8080);
  // You can also specify server:
  //Blynk.begin(auth, ssid, pass, "blynk-cloud.com", 8442);
  //Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8442);

  dht.begin();

  // Setup a function to be called every second
  timer.setInterval(1000L, sendSensor);
```

```
}

void loop()
{
  Blynk.run();
  timer.run();
  if ( Serial3.available() )   {
    Serial.write( Serial3.read() );
  }
  if ( Serial.available() )      {
    Serial3.write( Serial.read() );
  }
}
```

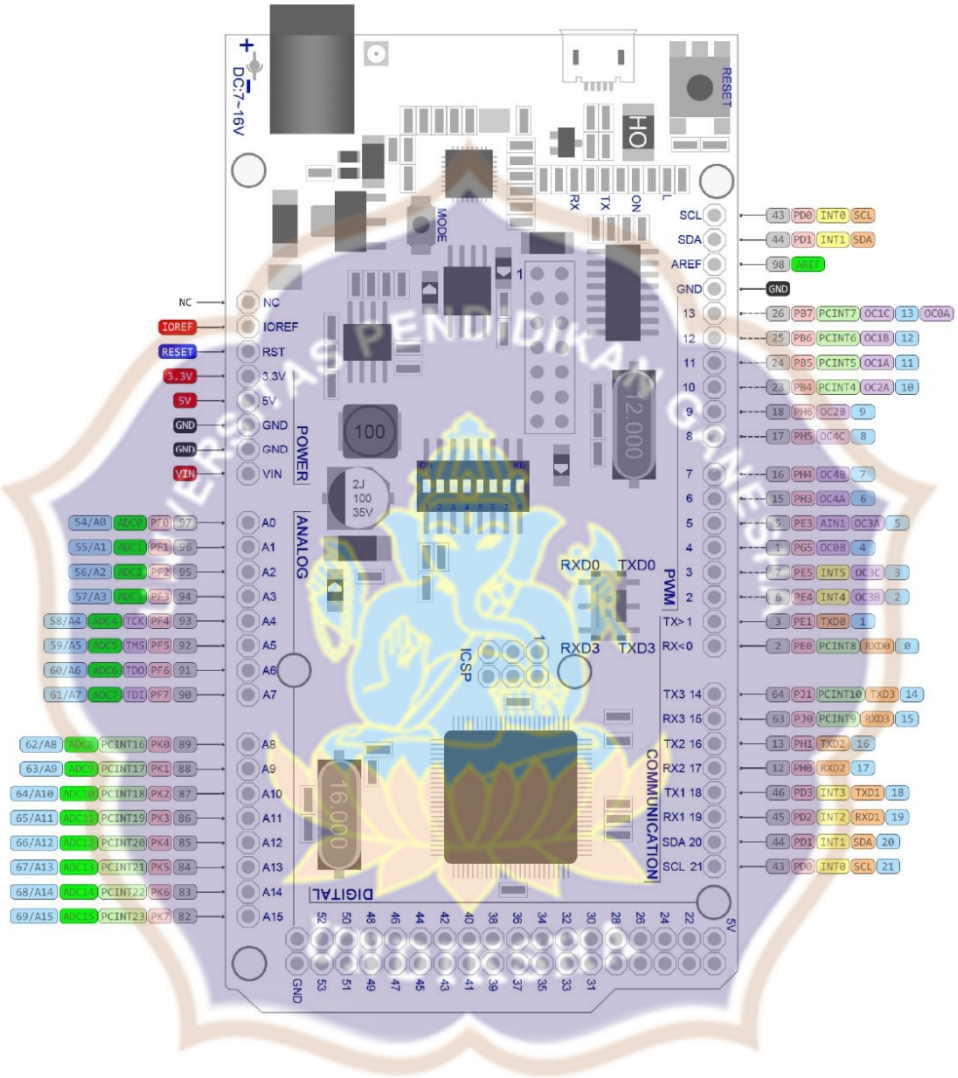
Lampiran 05

***Datasheet* Arduino Mega 2560 *with Built-in*-ESP8266**



RobotDyn®
www.robotdyn.com
PINOUT DIAGRAM

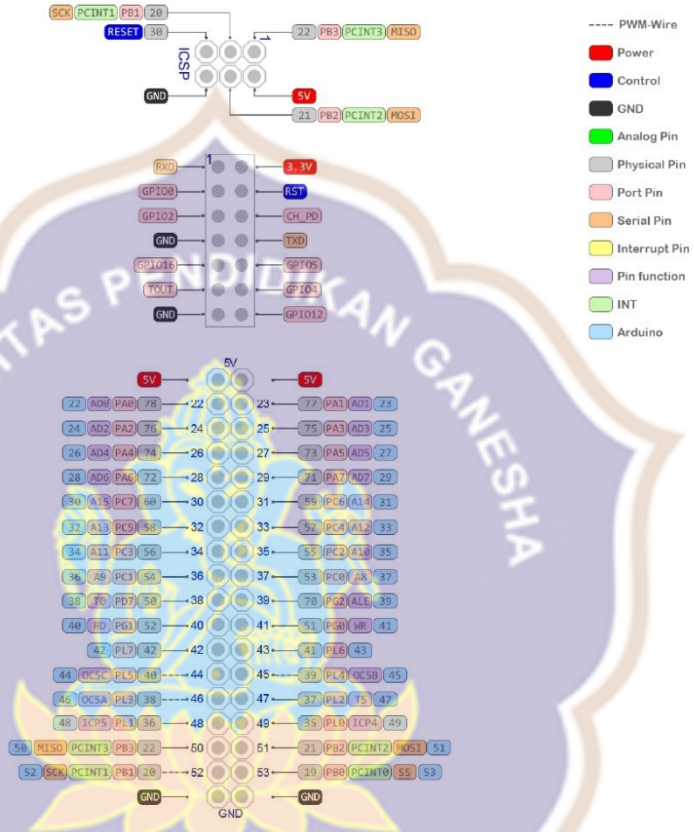
MEGA+WiFi R3 ATmega2560+ESP8266, flash 32MB, USB-TTL CH340G, Micro-USB



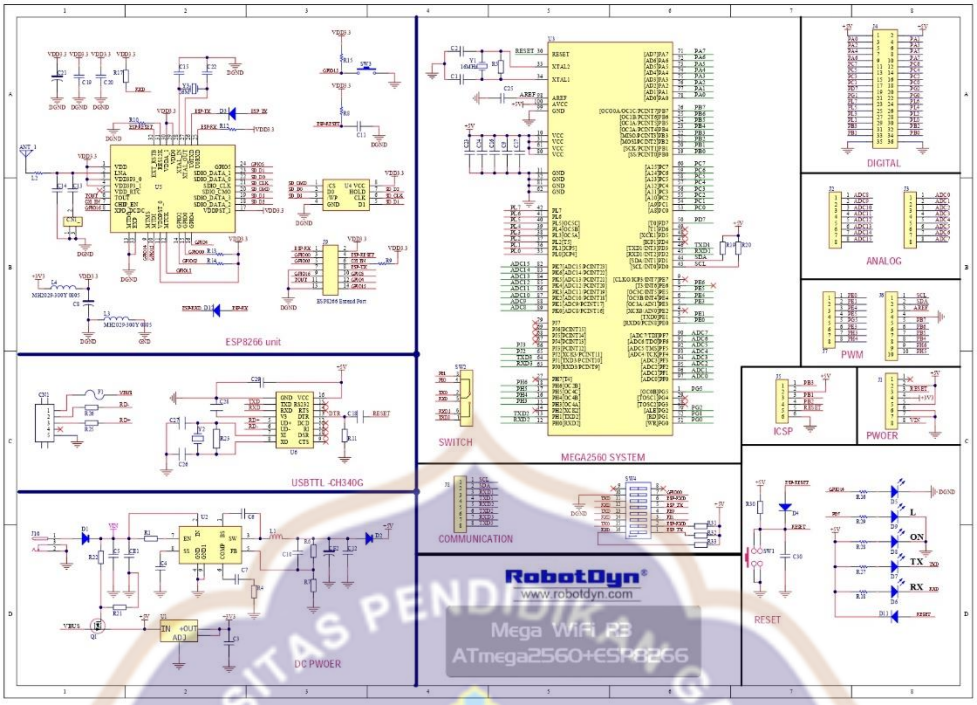
RobotDyn®
28 Jul 2017

RobotDyn®
www.robotdyn.com
PINOUT DIAGRAM

MEGA+WiFi R3 ATmega2560+ESP8266, flash 32MB, USB-TTL CH340G, Micro-USB



RobotDyn®
28 Jul 2017



Lampiran 06

***Datasheet* Motor Driver L298N**

HT **Handson Technology**

User Guide

L298N Dual H-Bridge Motor Driver

This dual bidirectional motor driver, is based on the very popular L298 Dual H-Bridge Motor Driver Integrated Circuit. The circuit will allow you to easily and independently control two motors of up to 2A each in both directions.It is ideal for robotic applications and well suited for connection to a microcontroller requiring just a couple of control lines per motor. It can also be interfaced with simple manual switches, TTL logic gates, relays, etc. This board equipped with power LED indicators, on-board +5V regulator and protection diodes.

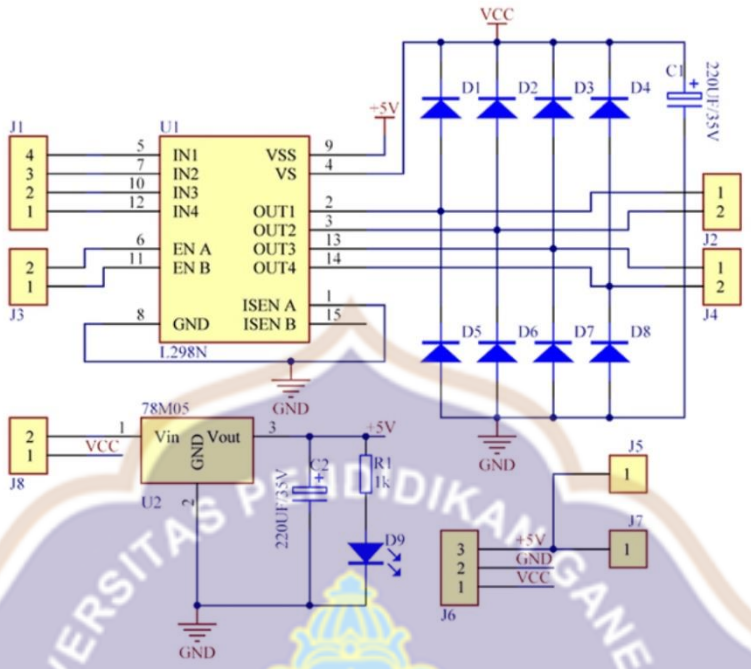
SKU: MDU-1049

Brief Data:

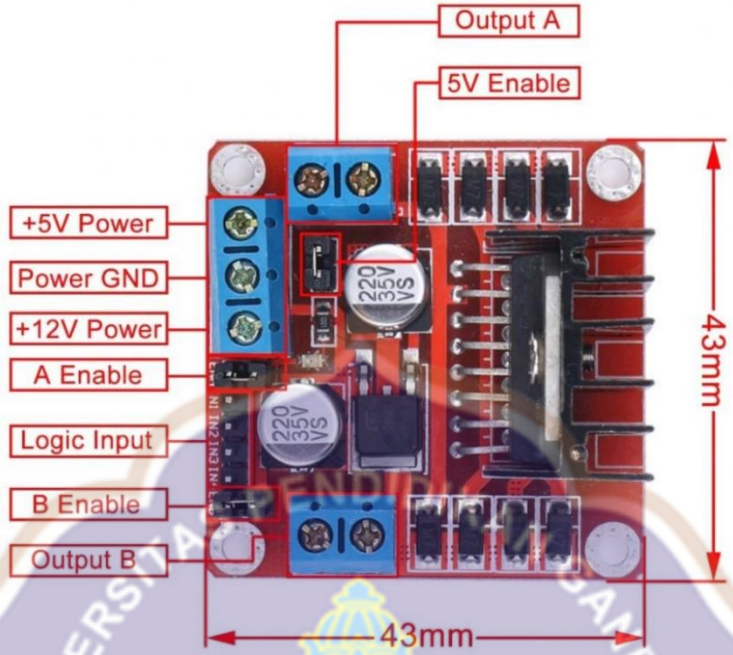
- Input Voltage: 3.2V~40Vdc.
- Driver: L298N Dual H Bridge DC Motor Driver
- Power Supply: DC 5 V - 35 V
- Peak current: 2 Amp
- Operating current range: 0 ~ 36mA
- Control signal input voltage range :
- Low: -0.3V ⩽ Vin ⩽ 1.5V.
- High: 2.3V ⩽ Vin ⩽ Vss.
- Enable signal input voltage range :
  - Low: -0.3 ⩽ Vin ⩽ 1.5V (control signal is invalid).
  - High: 2.3V ⩽ Vin ⩽ Vss (control signal active).
- Maximum power consumption: 20W (when the temperature T = 75 ℃).
- Storage temperature: -25 ℃ ~ +130 ℃.
- On-board +5V regulated Output supply (supply to controller board i.e. Arduino).
- Size: 3.4cm x 4.3cm x 2.7cm

1 www.handsontec.com

Schematic Diagram:



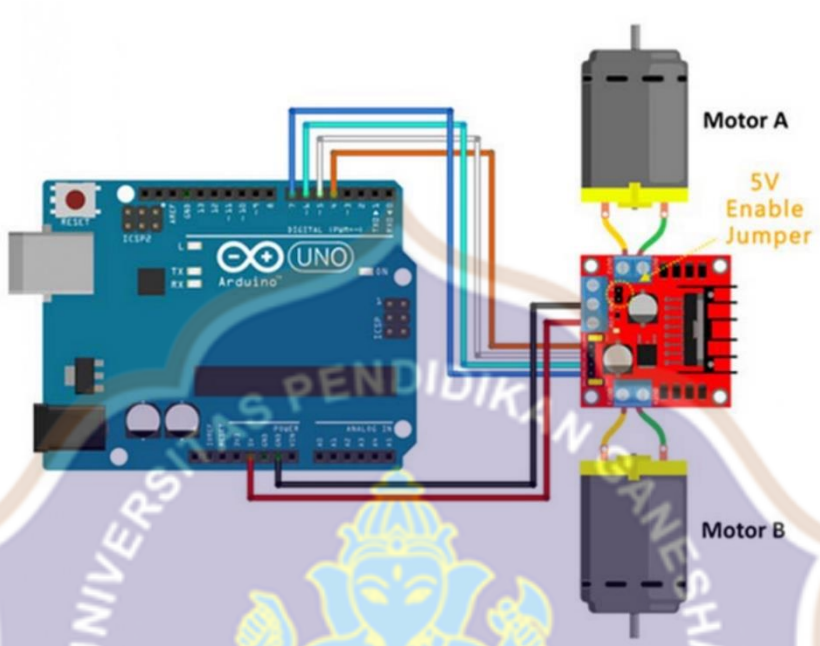
Board Dimension & Pins Function:



Connection Examples:

Controlling 2-DC Motor with +5V Arduino onboard Power Supply:

Below is the circuit connection use the on-board +5V power supply from Arduino board, and should be done without the 5V Enable Jumper on (Active 5V). This connection can drive two 5V DC motors simultaneously.



Sketch Listing:

Copy and paste the sketch below to Arduino IDE and upload to Arduino Uno/Mega board.

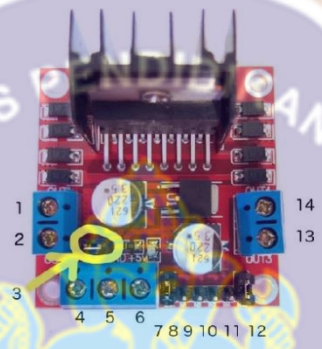
```
/*============================================================================
//  Author      : Handson Technology
//  Project     : Arduino Uno
//  Description : L298N Motor Driver
//  Source-Code : L298N_Motor.ino
//  Program: Control 2 DC motors using L298N H Bridge Driver
//============================================================================
*/

// Definitions Arduino pins connected to input H Bridge
int IN1 = 4;
int IN2 = 5;
int IN3 = 6;
int IN4 = 7;

void setup()
{
 // Set the output pins
```

```
 pinMode(IN1, OUTPUT);
 pinMode(IN2, OUTPUT);
 pinMode(IN3, OUTPUT);
 pinMode(IN4, OUTPUT);
}

void loop()
{
 // Rotate the Motor A clockwise
 digitalWrite(IN1, HIGH);
 digitalWrite(IN2, LOW);
 delay(2000);
 // Motor A
 digitalWrite(IN1, HIGH);
 digitalWrite(IN2, HIGH);
 delay(500);

 // Rotate the Motor B clockwise
 digitalWrite(IN3, HIGH);
 digitalWrite(IN4, LOW);
 delay(2000);
 // Motor B
 digitalWrite(IN3, HIGH);
 digitalWrite(IN4, HIGH);
 delay(500);

 // Rotates the Motor A counter-clockwise
 digitalWrite(IN1, LOW);
 digitalWrite(IN2, HIGH);
 delay(2000);
 // Motor A
 digitalWrite(IN1, HIGH);
 digitalWrite(IN2, HIGH);
 delay(500);

 // Rotates the Motor B counter-clockwise
 digitalWrite(IN3, LOW);
 digitalWrite(IN4, HIGH);
 delay(2000);
 // Motor B
 digitalWrite(IN3, HIGH);
 digitalWrite(IN4, HIGH);
 delay(500);
}
```

Controlling Stepper Motor

In this example we have a typical NEMA-17 stepper motor with four wires:

The key to successful stepper motor control is identifying the wires - that is which one is which. You will need to determine the A+, A-, B+ and B- wires. With our example motor these are red, green, yellow and blue. Now let's get the wiring done.



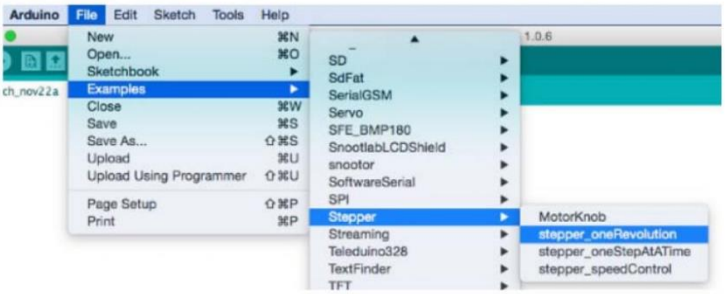
Connect the A+, A-, B+ and B- wires from the stepper motor to the module connections 1, 2, 13 and 14 respectively. Place the jumpers included with the L298N module over the pairs at module points 7 and 12. Then connect the power supply as required to points 4 (positive) and 5 (negative/GND).

Once again if your stepper motor's power supply is less than 12V, fit the jumper to the module at point 3 which gives you a neat 5V power supply for your Arduino.

Next, connect L298N module pins IN1, IN2, IN3 and IN4 to Arduino digital pins D8, D9, D10 and D11 respectively. Finally, connect Arduino GND to point 5 on the module, and Arduino 5V to point 6 if sourcing 5V from the module.

Controlling the stepper motor from your sketches is very simple, thanks to the *Stepper* Arduino library included with the Arduino IDE as standard.

To demonstrate your motor, simply load the *"stepper_oneRevolution"* sketch that is included with the *Stepper* library, for example:



Finally, check the value for

```
const int stepsPerRevolution = 200;
```

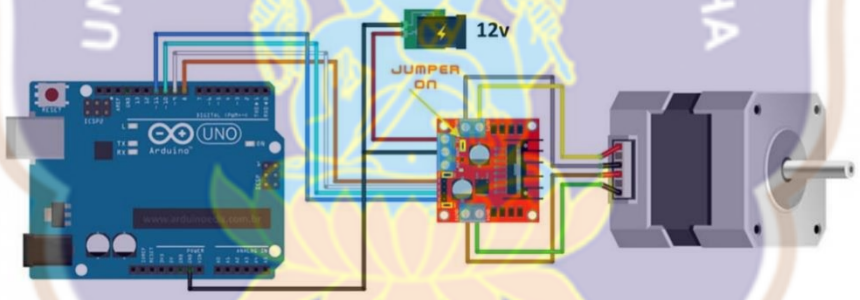
in the sketch and change the 200 to the number of steps per revolution for your stepper motor, and also the speed which is preset to 60 RPM in the following line:

```
myStepper.setSpeed(60);
```

Now you can save and upload the sketch, which will send your stepper motor around one revolution, then back again. This is achieved with the function

```
myStepper.step(stepsPerRevolution); // for clockwise
myStepper.step(-stepsPerRevolution); // for anti-clockwise
```

Connection for the sketch "*stepper_oneRevolution*":



Web Resources:

Lampiran 07

***Datasheet* Sensor Suhu dan Kelembaban DHT11**

SUNROM TECHNOLOGIES

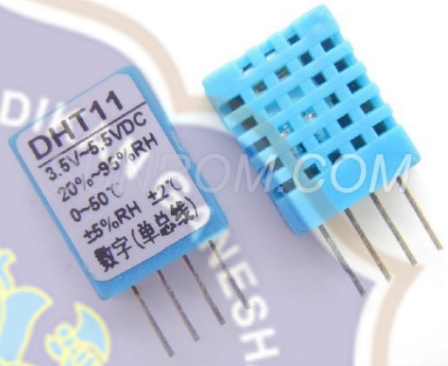
For more products visit our website http://www.sunrom.com

Document: Datasheet Date: 20-Jun-12 Model #: 3732 Product's Page: www.sunrom.com/p-1141.html

DHT11 - Humidity and Temperature Sensor

The DHT11 is a basic, low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed).

Its fairly simple to use, but requires careful timing to grab data. The only real downside of this sensor is you can only get new data from it once every 2 seconds.



Features

- Full range temperature compensated
- Relative humidity and temperature measurement
- Calibrated digital signal
- Outstanding long-term stability
- Extra components not needed
- Long transmission distance
- Low power consumption
- 4 pins packaged and fully interchangeable

Details

This sensor includes a resistive-type humidity measurement component and an NTC temperature measurement component, and connects to a high-performance 8-bit microcontroller, offering excellent quality, fast response, anti-interference ability and cost-effectiveness. Each DHT11 element is strictly calibrated in the laboratory that is extremely accurate on humidity calibration. The calibration coefficients are stored as programmes in the OTP memory, which are used by the sensor's internal signal detecting process.
The single-wire serial interface makes system integration quick and easy. Its small size, low power consumption and up-to-20 meter signal transmission making it the best choice for various applications, including those most demanding ones. The component is 4-pin single row pin package.

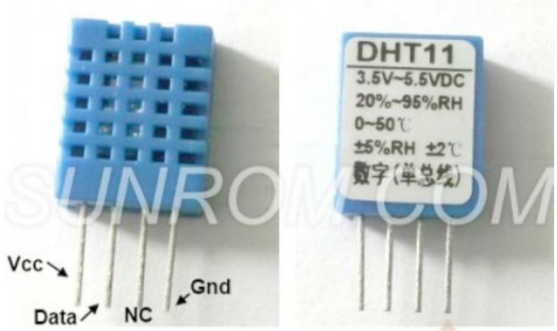
Specifications

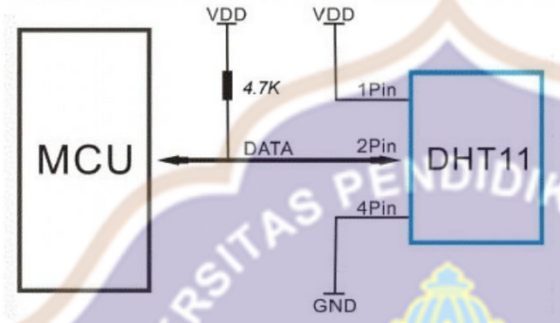
Item	Measurement Range	Humidity Accuracy	Temperature Accuracy	Resolution	Package
DHT11	20-90%RH 0-50 ℃	±5%RH	±2℃	1	4 Pin Single Row

Parameters	Conditions	Minimum	Typical	Maximum
Humidity				
Resolution		1%RH	1%RH	1%RH
			8 Bit	
Repeatability			±1%RH	
Accuracy	25℃		±4%RH	
	0-50℃			±5%RH
Interchangeability	Fully Interchangeable			
Measurement Range	0℃	30%RH		90%RH
	25℃	20%RH		90%RH
	50℃	20%RH		80%RH
Response Time (Seconds)	1/e(63%)25℃, 1m/s Air	6 S	10 S	15 S
Hysteresis			±1%RH	
Long-Term Stability	Typical		±1%RH/year	
Temperature				
Resolution		1℃	1℃	1℃
		8 Bit	8 Bit	8 Bit
Repeatability			±1℃	
Accuracy		±1℃		±2℃
Measurement Range		0℃		50℃
Response Time (Seconds)	1/e(63%)	6 S		30 S

Item	Condition	Min	Typical	Max	Unit
Power supply	DC	3	5	5.5	V
Current supply	Measuring	0.5		2.5	mA
	Stand-by	100	Null	150	uA
	Average	0.2	Null	1	mA

Typical Application





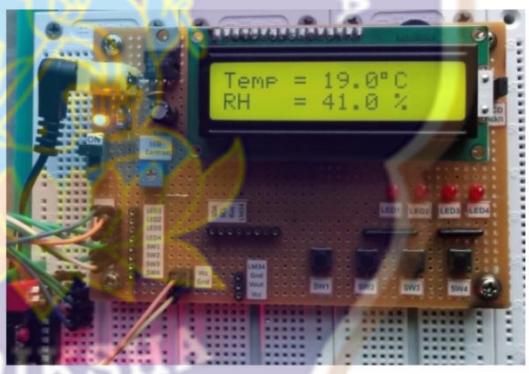
DHT11's power supply is 3-5.5V DC. When power is supplied to the sensor, do not send any instruction to the sensor in within one second in order to pass the unstable status. One capacitor valued 100nF can be added between VDD and GND for power filtering.

SDK (Software Development Kit)

Download source code + project articles by clicking following link

http://www.sunrom.com/files/3732.zip

It contains details for AVR, PIC and Arduino projects.



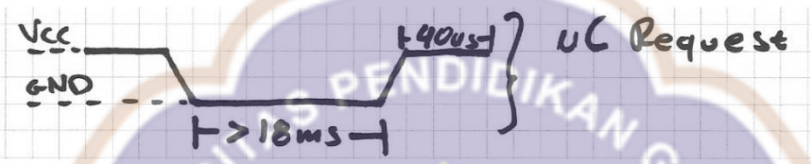
Communication Process: Serial Interface (Single-Wire Two-Way)

The interesting thing in this module is the protocol that uses to transfer data. All the sensor readings are sent using a single wire bus which reduces the cost and extends the distance. In order to send data over a bus you have to describe the way the data will be transferred, so that transmitter and receiver can understand what says each other. This is what a protocol does. It describes the way the data are transmitted. On DHT-11 the 1-wire data bus is pulled up with a resistor to VCC. So if nothing is occurred the voltage on the bus is equal to VCC.

Communication Format can be seperated into three stages

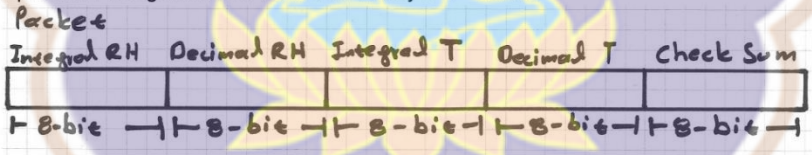
1) Request
2) Response
3) Data Reading

1) **Request:** To make the DHT-11 to send you the sensor readings you have to send it a request. The request is, to pull down the bus for more than **18ms** in order to give DHT time to understand it and then pull it up for **40uS**.



2) **Response:** What comes after the request is the DHT-11 response. This is an automatic reply from DHT which indicates that DHT received your request. The response is ~54uS low and 80uS high.

3) **Data Reading:** What will come after the response is the sensor data. The data will be packed in a packet of 5 segments of 8-bits each. Totally 5×8 =40bits.



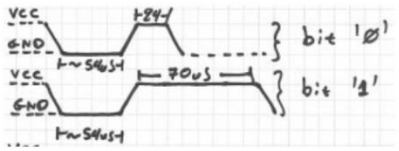
First two segments are Humidity read, integral & decimal. Following two are Temperature read in Celsius, integral & decimal and the last segment is the Check Sum which is the sum of the 4 first

segments. If Check Sum's value isn't the same as the sum of the first 4 segments that means that data received isn't correct.

How to Identify Bits: Each bit sent is a follow of ~54uS Low in the bus and ~24uS to 70uS High depending on the value of the bit.

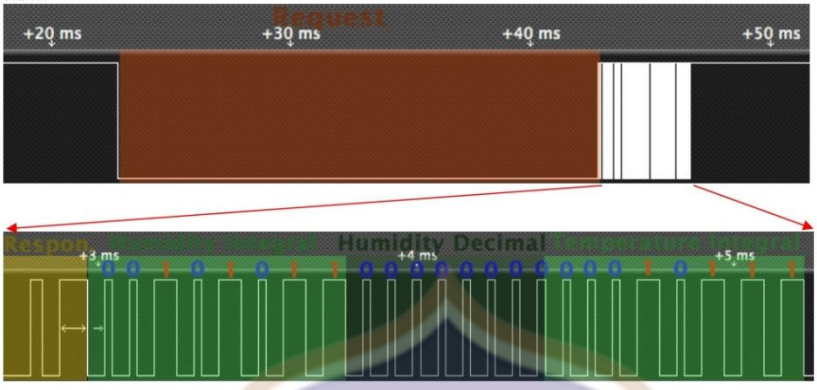
Bit '0' : ~54uS Low and ~24uS High

Bit '1' : ~54uS Low and ~70uS High



End Of Frame: At the end of packet DHT sends a ~54uS Low level, pulls the bus to High and goes to sleep mode.

Logic Analyzer Snapshots: In the following image you can see the request sent from the MCU to the DHT and following the packet. Because the request has very long duration as you can see is about 20mS and packet received is in uS we can't view the data bits. So it is exapanded in next view.



If we zoom at the data bits we can read the values. You can see after the Request follows the Response, and Data bits. I have drawn some color notes to be more understandable.

If we decode the above data we have.

Humidity 0b00101011.0b00000000 = 43.0% (43 is integral part and .0 is decimal part)

Temperature 0b00010111 = 23 C.

The last two segments can't be seen in this image because of zoom.

Implementation:

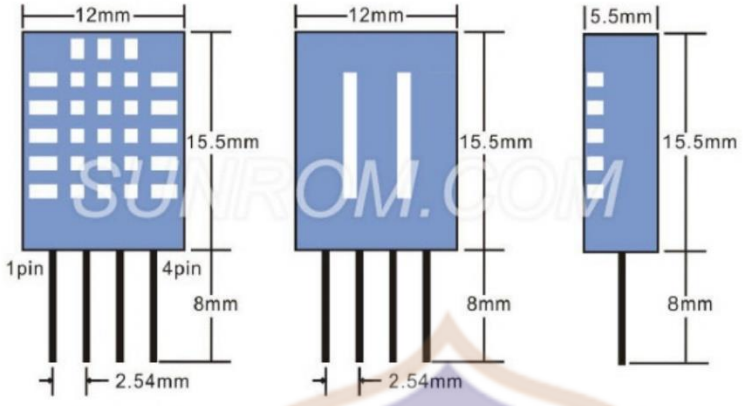
What we have to do to read a DHT-11 sensor is:

1) Send request
2) Read response
3) Read each data segment and save it to a buffer
4) Sum the segments and check if the result is the same as CheckSum

If the CheckSum is correct, the values are correct so we can use them. If CheckSum is wrong we discard the packet.

To read the data bits can use a counter and start count uSeconds of High level. For counts > 24uS we replace with bit '1'. For counts <=24 we replace with bit'0'

Dimensions (mm)

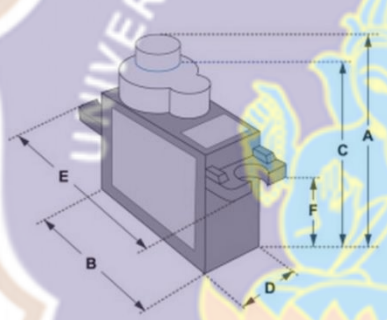


Lampiran 08

***Datasheet* Servo SG90**

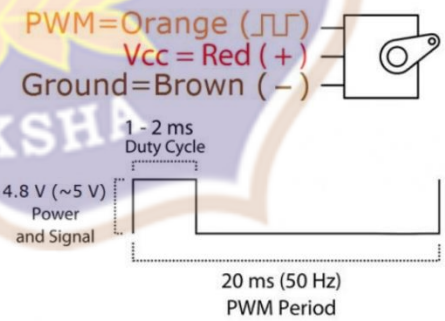
SERVO MOTOR SG90 DATA SHEET

Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control these servos. Good 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. It comes with a 3 horns (arms) and hardware.



Dimensions & Specifications
A (mm) : 32
B (mm) : 23
C (mm) : 28.5
D (mm) : 12
E (mm) : 32
F (mm) : 19.5
Speed (sec) : 0.1
Torque (kg-cm) : 2.5
Weight (g) : 14.7
Voltage : 4.8 - 6

Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is middle, is all the way to the right, "-90" (~1ms pulse) is all the way to the left.



Lampiran 09

Dokumentasi