1. Introduction

The ILI9488 is a 16.7M single-chip SoC driver for a-Si TFT liquid crystal display panels with a resolution of 320(RGB) x 480 dots. The ILI9488 is comprised of a 960-channel source driver, a 480-channel gate driver, 345,600 bytes GRAM for graphic data of 320 (RGB) x 480 dots, and power supply circuit.

The ILI9488 supports parallel DBI Type B 8-/9-/16-/18-/24-bit data bus interfaces and DBI Type C 3-/4-line serial peripheral interfaces (SPI) to input commands. The ILI9488 supports DPI (16-/18-/24-bit) data bus for video image display. For MIPI*-DSI* high-speed interface mode, the ILI9488 also provides one data lane and one clock lane that can support up to 500Mbps on MIPI-DSI link.

The ILI9488 can operate with 1.65V I/O interface voltage and supports a wide range of analog power supplies. The ILI9488 supports 8-colors display and sleep mode power management functions, ideal for portable products where battery power conservation is desirable, such as digital cellular phones, smart phones, MP3 players, personal media players and similar devices with color graphics displays.

Notes:

- MIPI: Mobile Industry Processor Interface
- DSI: Display Serial Interface

2. Features

- Display resolution: 320 (RGB) (H) x 480 (V)
- Display color modes:
  - Full color modes:
    - 16.7M colors with dithering function (24-bit data, R: 8-bit, G: 8-bit, B: 8-bit)
    - 262K colors (18-bit data, R: 6-bit, G: 6-bit, B: 6-bit)
  - Reduced color modes:
    - 65K colors (16-bit data, R: 5-bit, G: 6-bit, B: 5-bit)
    - 8 colors (3-bit data, R: 1-bit, G: 1-bit, B: 1-bit)
- Display module:
  - On-chip Frame Memory size 345,600 bytes, 320 (RGB) (H) x 480 (V) x 18 bits
  - Supports 960 source channel outputs
  - Supports up to a maximum of 480 gate lines
  - Supports 24-bits input image function
  - Supports column/1-/2-dot inversion
  - On-module DC VCOM control (-2 to 0V common electrode output voltage range)
  - Source/VCOM/Gate power supply voltage
    - DDVDH – GND = 4.5 to 6V
    - DDVDL – GND = -6 to -4.5V
    - VCL – GND = -3 to -2V
    - DC VCOM – GND = -2 to 0V, a step = 16mV
    - VREG1OUT – GND = 3.625 to 5.5 V
    - VREG2OUT – GND = -5.5 to -3.625V
    - VGH – GND = 10 to 20V
    - VGL – GND = -15 to -5V
- Display Interface types:
  - MIPI-DBI (Display Bus Interface)
    - Type B (i-80 system), 8-/9-/16-/18-/24-bit bus
    - Type C (Serial data transfer interface, 3/4-line SPI)
  - MIPI-DPI (Display Pixel Interface)
    - Supports 24 bit/pixel (R: 8-bit, G: 8-bit, B: 8-bit)
    - Supports 18 bit/pixel (R: 6-bit, G: 6-bit, B: 6-bit)
    - Supports 16 bit/pixel (R: 5-bit, G: 6-bit, B: 5-bit)
  - MIPI-DSI (Display Serial Interface)
    - Supports one data lane/maximum speed 500Mbps
    - Supports DSI version 1.01
    - Supports D-PHY version 1.00
- Input power
  - Low operating power supplies
    - IOVCC = 1.65 to 3.3V (Interface I/O)
    - VCI = 2.5 to 3.3V (Analog)
    - OTP programming voltage (DDVDH) = 7V
- Power saving modes:
  - Deep-standby mode
  - Sleep mode

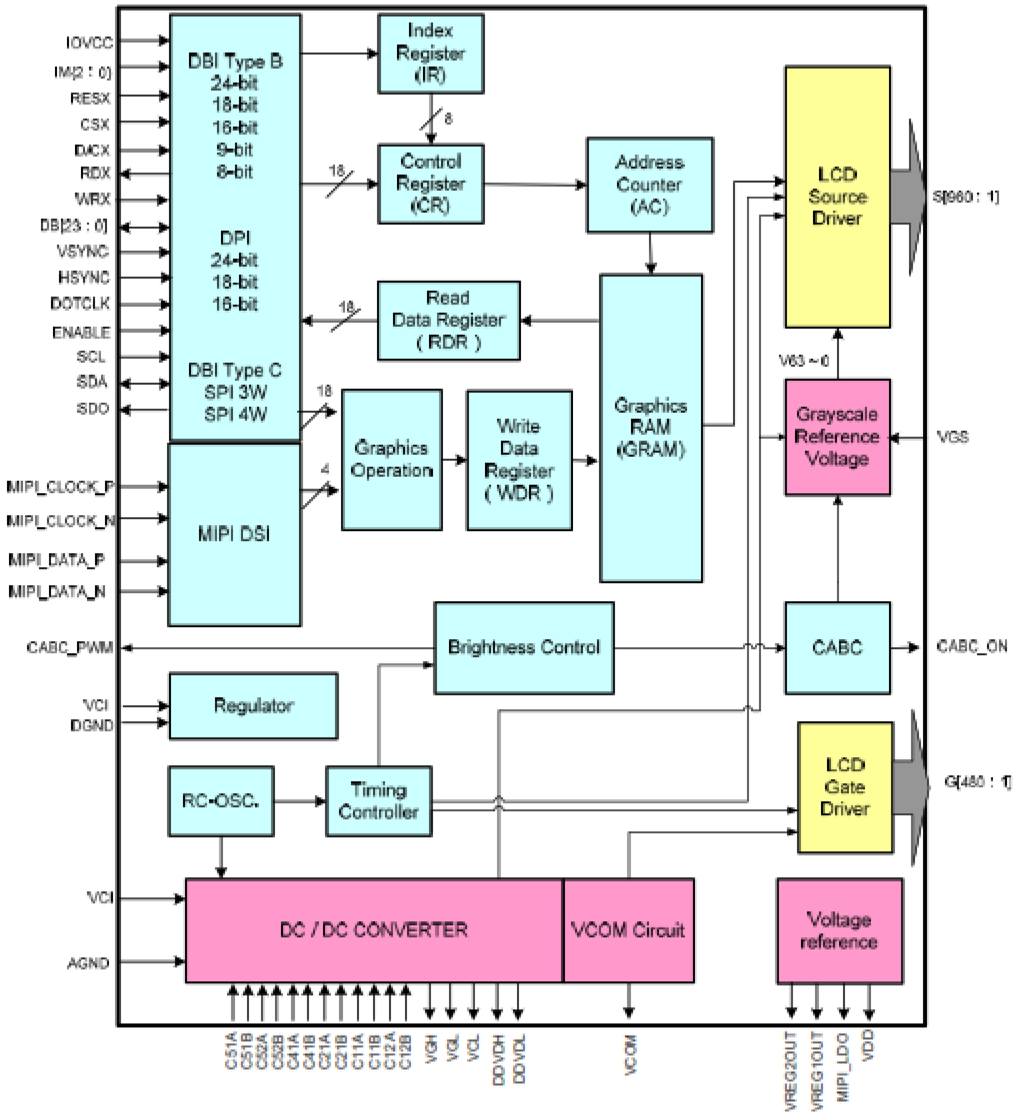
- Other on-chip functions/Miscellaneous
  - Supports partial display mode
  - Supports inversion mode
  - Oscillator for display clock generation
  - LVD function (GAS bit) prevents image sticking for abnormal power off
  - Supports DC VCOM driving
  - DC VCOM voltage generator and adjustment
  - OTP memory store initialization register settings (MATCDL, VRH1, VRH2 and BT)
  - MTP (provides 4 times OTP to store DC VCOM setting, ID1/ID2/ID3 setting)
  - Supports CABC function
  - Supports 3-Gamma DGC function
  - Supports dither function. (The dither function is only available in Bypass mode of DPI 24-bit and MIPI-DSI.)
  - Supports color enhancement function

- Operate temperature range: -40℃ to 85℃

Notes:
- CABC: Content Adaptive Brightness Control
- DGC: Digital Gamma Correction
- LVD: Low Voltage Detection
- MTP: Multiple Time Programming
- OTP: One Time Programming

3. Device Overview

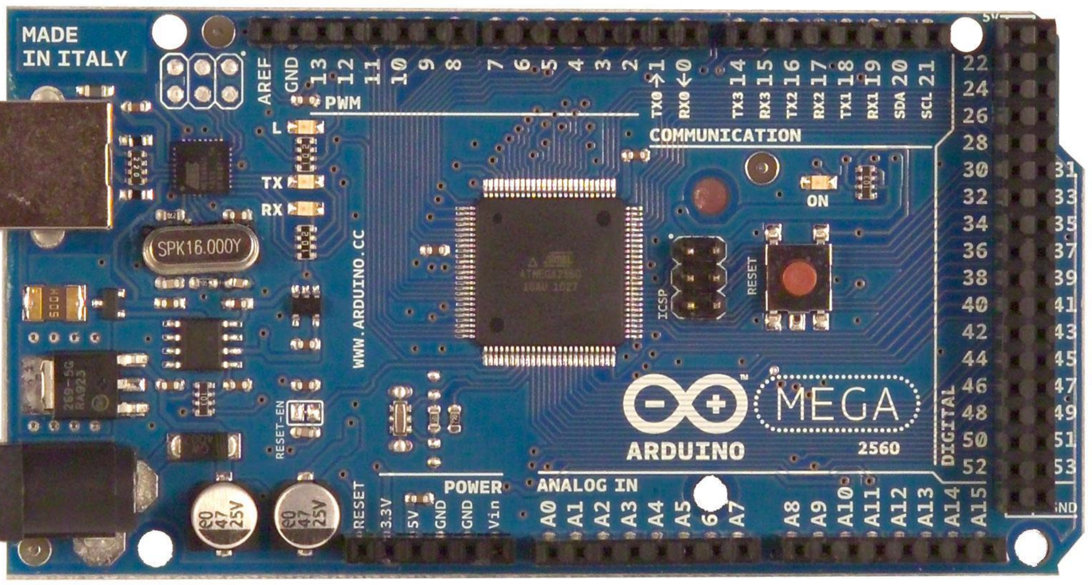
3.1. Block Diagram

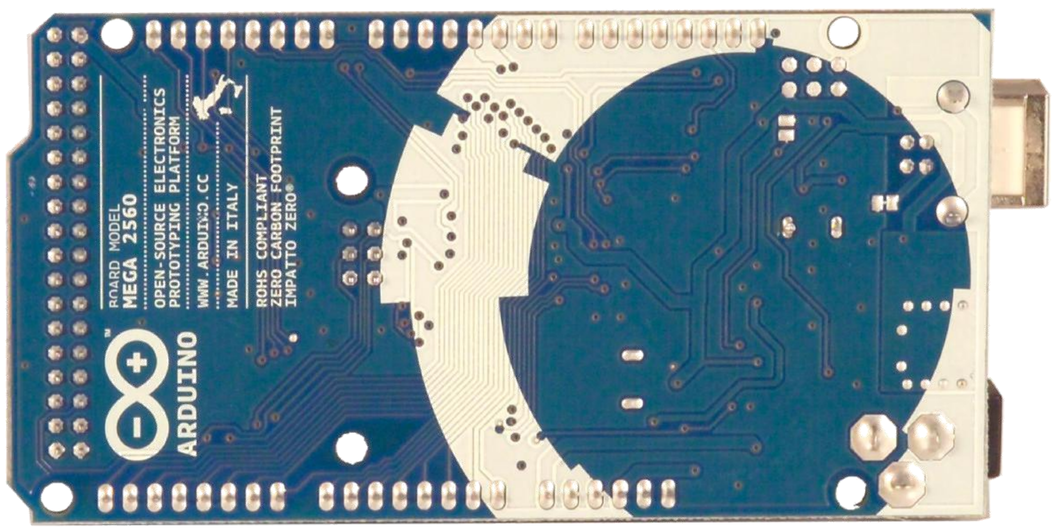


www.robotshop.com

La robotique à votre service! - Robotics at your service!

Arduino Mega 2560 Datasheet


MADE IN ITALY
AREF GND 13 12 11 10 9 8 7 6 5 4 3 2 TX0→1 RX0←0 TX3 14 RX3 15 TX2 16 RX2 17 TX1 18 RX1 19 SDA 20 SCL 21
PWM
COMMUNICATION
L
TX
RX
ON
SPK16.000Y
WWW.ARDUINO.CC
ICSP
RESET
RESET-EN
ARDUINO
MEGA
2560
DIGITAL
POWER
ANALOG IN
RESET 3.3V 5V GND GND Vin
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15



Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, 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 Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Schematic & Reference Design

EAGLE files: arduino-mega2560-reference-design.zip

Schematic: arduino-mega2560-schematic.pdf

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Power

The Arduino Mega 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 Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

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.** The regulated power supply used to power the microcontroller and othercomponents on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is50 mA.
- **GND.** Ground pins.

Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

-

Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode()

, digitalWrite(),anddigitalRead() functions.They operate at 5 volts. Each pin can provide orreceive 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); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX)**

  and 16 (TX); Serial 3: 15 (RX) and 14 (TX). Used to receive (RX) and transmit(TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.

- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5),**

  19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2). These pins can beconfigured 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: 0 to 13.** Provide 8-bit PWM output with theanalogWrite()function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPIcommunication using the SPI library. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Uno, Duemilanove and Diecimila.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGHvalue, the LED is on, when the pin is LOW, it's off.

- **I$_2$C: 20 (SDA) and 21 (SCL).** Support I$_2$C (TWI) communication using theWirelibrary(documentationon the Wiring website). Note that these pins are not in thesame location as the I$_2$C pins on the Duemilanove or Diecimila.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used withanalogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a resetbutton to shields which block the one on the board.

Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Mega2560's digital pins.

The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiringwebsitefordetails. For SPI communication, use theSPI library.

Programming

The Arduino Mega can be programmed with the Arduino software (download). For details, see the referenceand tutorials.

The ATmega2560 on the Arduino Mega comes preburned with a bootloaderthat allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see theseinstructions for details.

Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Arduino Mega2560 is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega2560 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Mega2560 is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Mega2560. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened.

If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Mega2560 contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

The Arduino Mega2560 has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics and Shield Compatibility

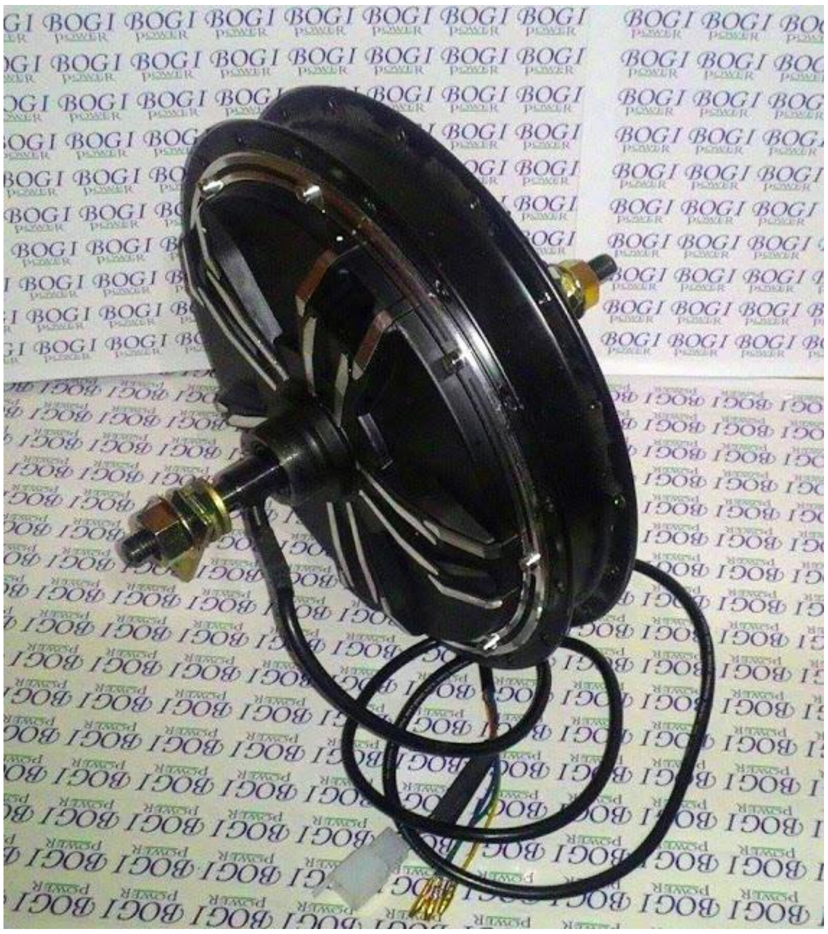
The maximum length and width of the Mega2560 PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

The Mega2560 is designed to be compatible with most shields designed for the Uno, Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Further the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega2560 and Duemilanove / Diecimila. *Please note that I2C is not located on thesame pins on the Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5)*

MOTOR BLDC 350W HI TORSI

Spesifikasi Electric

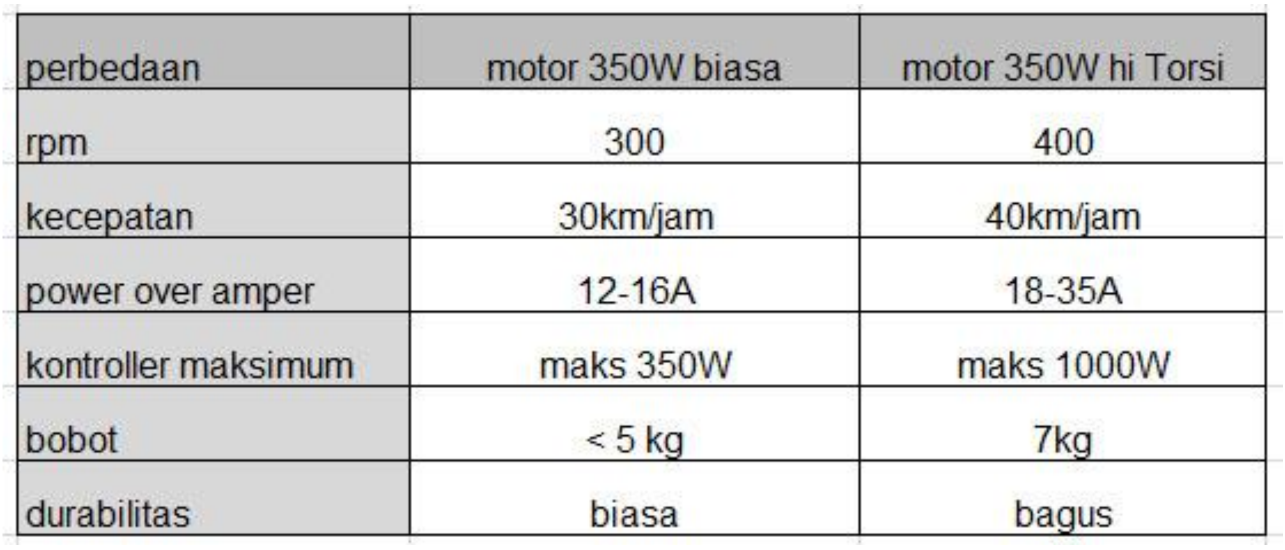
Tegangan	=	48V
Power watt	=	350W
over power watt	=	> 1000W
Amper kerja	=	16-18A
OverAmper maks	=	>35A
Torsi	=	18-25Nm
recomended kontroler	=	48V 350W 17A full fitur kontroller
maksimum kontroler	=	48V 1000W 35A full fitur kontroller



Spesifikasi Model

Model socket = skun bulat lonjong (Male) , socket O ring. atau by request
Socket hall = socket 6 pin isi 5. (Male)
Jumlah ruji = 36 lubang
Compatible Rim velg = 20", 24", 26" 700c, dan 17" ring sepeda motor
open size as = 15 cm. panjang as = 18cm
diameter motor = 24 cm
lebar motor = 7 cm
bobot = 6 Kg.
Sistem rem = flaksible (terdapat drat untuk adapter tromol maupun disk brake)
Kecepatan = 36V: 35 km/jam, 48V: 45km/jam

PERBEDAAN STANDAR DAN HI TORSI



perbedaan	motor 350W biasa	motor 350W hi Torsi
rpm	300	400
kecepatan	30km/jam	40km/jam
power over amper	12-16A	18-35A
kontroller maksimum	maks 350W	maks 1000W
bobot	< 5 kg	7kg
durabilitas	biasa	bagus

BRAND XS MOTOR.

DESAIN TERBARU , AGUSTUS 2016

APLIKASI: Sepeda listrik model Dowhill, Becak Listrik, Mobil Wahana listrik, Mobil listrik model Proto / Urban concept.
Recomended kontroller:

BLDC 350W full fitur,

BLDC 350W Sensorless dual mode

Stock Tersedia **:** **READY STOCK >10 PCS**
Harga baru : **call CS**

PROGRAM KURSI RODA ELEKTRIK

BERBASIS ARDUINO MEGA 2560

```
// URTouch_ButtonTest
// This program is a quick demo of how create and use buttons.
// This program requires the UTFT library.
// It is assumed that the display module is connected to an
// appropriate shield or that you know how to change the pin
// numbers in the setup.
#include <UTFT.h>
#include <URTouch.h>
// Define the orientation of the touch screen. Further
// information can be found in the instructions.
#define TOUCH_ORIENTATION  PORTRAIT
// Initialize display
// ------------------
// Set the pins to the correct ones for your development board
// -----------------------------------------------------------
// Standard Arduino Uno/2009 Shield            : <display model>,19,18,17,16
// Standard Arduino Mega/Due shield            : <display model>,38,39,40,41
// CTE TFT LCD/SD Shield for Arduino Due       : <display model>,25,26,27,28
// Teensy 3.x TFT Test Board                   : <display model>,23,22, 3, 4
// ElecHouse TFT LCD/SD Shield for Arduino Due : <display model>,22,23,31,33
//
// Remember to change the model parameter to suit your display module!
UTFT myGLCD(ITDB32WC,38,39,40,41);
// Initialize touchscreen
// ----------------------
// Set the pins to the correct ones for your development board
// -----------------------------------------------------------
// Standard Arduino Uno/2009 Shield            : 15,10,14, 9, 8
// Standard Arduino Mega/Due shield            :  6, 5, 4, 3, 2
// CTE TFT LCD/SD Shield for Arduino Due       :  6, 5, 4, 3, 2
// Teensy 3.x TFT Test Board                   : 26,31,27,28,29
// ElecHouse TFT LCD/SD Shield for Arduino Due : 25,26,27,29,30
//
URTouch  myTouch( 6, 5, 4, 3, 2);
// Declare which fonts we will be using
extern uint8_t BigFont[];
extern uint8_t SevenSegNumFont[];
uint32_t cx, cy;
uint32_t rx[8], ry[8];
uint32_t clx, crx, cty, cby;
float px, py;
int dispx, dispy, text_y_center;
uint32_t calx, caly, cals;
char buf[13];
bool setnewcode_flag = 0;
bool setoldcode_flag = 0;
bool repeat_flag = 1;
bool lock_flag = 0;
int x, y, z;
String stCode = "12345";
String stTemp = "";
char stCurrent[6] = ""; //array 6 dimensi untuk menyimpan inputan angka bukan password
char asterisc[6] = ""; // hidden password dengan bentuk bintang - bintang
char asterisc_partial[6] = "";
String stLast = "";
String msgCurrent  = "";//pesan blink tulisan ke lcd
int stCurrentLen=0; // menghapus memori sementara
int thisByte = 0; // data serial hexa desimal
int data_sent = 0; // data yang di kirim
int buttonState = 0; //kondisi awal button adalah 0
int SW_pin = 8; // brake
int sensorPin = A0;    // select the input pin for the photodioda
int sensorValue = 0;  // variable to store the value coming from the sensor
const int X_pin = 9; // analog pin connected to X output
const int Y_pin = 10; // analog pin connected to Y output
int kiri1 = 10; // kecepatan motor
int kanan1 = 9;
int LR = 11; // rotasi motor
int RR = 12;
int gigimundur = 18;
int power = 13;
int nilai;
int gigimundurState = 0;
int powerState = 0;
int count=0;
int val;// simpan nilai kesalahan
int stLastserial;
/*************************
**   Custom functions   **
*************************/
void cleardisplay () //kotak hapus angka inputan
{
  myGLCD.setColor(230, 230, 230);
  //myGLCD.fillRect(0, 0, 239, 50);      // clear display
  myGLCD.fillRect(30, 230, 285, 190);      // clear display untuk kotak besar //fix
}
void clearsmalldisplay () //menghapus angka pwd dan tanda bintang
{
  myGLCD.setColor(255, 255, 255);
  //myGLCD.fillCircle(51, 27, 16);
  myGLCD.fillCircle(120, 210, 16);  //set lingkaran kecil tempat pwd kiri // FIX
  //myGLCD.fillCircle(188, 27, 16);
  myGLCD.fillCircle(200, 210, 16); //set lingkaran kecil tempat pwd kanan //FIX
  //myGLCD.fillRect(51, 11, 188, 43);    // clear small display
  myGLCD.fillRect(120, 225, 200, 195);    // clear small display //FIX
}
void drawButtons()
{
// Draw the upper row of buttons
```

```
  for (x=0; x<5; x++)
  {
    myGLCD.setColor(0, 0, 255);
    myGLCD.fillRoundRect (0+(x*100), 0, 60+(x*100), 60);
    myGLCD.setColor(255, 255, 255);
    myGLCD.drawRoundRect (0+(x*100), 0, 60+(x*100), 60);
    myGLCD.printNumI(x+1, 27+(x*100), 27);
  }
// Draw the center row of buttons
  for (x=0; x<5; x++)
  {
    myGLCD.setColor(0, 0, 255);
    myGLCD.fillRoundRect (0+(x*100), 80, 60+(x*100), 140);
    myGLCD.setColor(255, 255, 255);
    myGLCD.drawRoundRect (0+(x*100), 80, 60+(x*100), 140);
    if (x<4)
      myGLCD.printNumI(x+6, 27+(x*100), 107);
  }
  myGLCD.print("0", 427, 107);
// Draw the lower row of buttons
  myGLCD.setColor(0, 0, 255);
  myGLCD.fillRoundRect (0, 160, 230, 240);
  myGLCD.setColor(255, 255, 255);
  myGLCD.drawRoundRect (0, 160, 230, 240);
  myGLCD.print("CLEAR", 115, 190);
  myGLCD.setColor(0, 0, 255);
  myGLCD.fillRoundRect (240, 160, 460, 240);
  myGLCD.setColor(255, 255, 255);
  myGLCD.drawRoundRect (240, 160, 460, 240);
  myGLCD.print("ENTER", 365, 190);
  myGLCD.setBackColor (0, 0, 0);
}
void blinkmsg (String msg, String color) //blok membuat tulisan dan warna
{
  myGLCD.setBackColor (290, 290, 290);
  if (color == "RED")
  {
    myGLCD.setColor(255, 0, 0);
  }
  if (color == "WHITE")
  {
    myGLCD.setColor(255, 255, 255);
  }
  if (color == "GREEN")
  {
    myGLCD.setColor(0, 255, 0);
  }
  if (color == "BLACK")
  {
    myGLCD.setColor(0, 0, 0);
  }
  myGLCD.print(msg, CENTER, 290);              // blink msg
  delay(500);
  myGLCD.print("              ", CENTER, 290);
  delay(500);
  myGLCD.print(msg, CENTER, 290);
  delay(500);
  myGLCD.print("              ", CENTER, 290);
  delay(500);
}
void printmsg (String msg, String color) //blok bagian menampilkan
{
  if (color == "WHITE")
  {
    myGLCD.setBackColor (0, 0, 0);
    myGLCD.setColor(255, 255, 255);
  }
  if (color == "GREEN")
  {
    myGLCD.setBackColor (290, 290, 290);
    myGLCD.setColor(0, 255, 0);
  }
  if (color == "RED")
  {
    myGLCD.setBackColor (290, 290, 290);
    myGLCD.setColor(255, 0, 0);
  }
  if (color == "BLACK")
  {
    myGLCD.setBackColor (290, 290, 290);
    myGLCD.setColor(0, 0, 0);
  }
    myGLCD.print(msg, CENTER, 290);
}
    void updateStr(int val)
{
  if (stCurrentLen<5) //sesuikan dengan jumlah pwd
  {
    if (stCurrentLen==0)
    {
      cleardisplay();
      clearsmalldisplay();
    }
    else
    clearsmalldisplay();
    stCurrent[stCurrentLen]=val;
    stCurrent[stCurrentLen+1]='\0';
    asterisc[stCurrentLen]='*';
    asterisc[stCurrentLen+1]='\0';
    if (stCurrentLen>0)
    {
      asterisc_partial[stCurrentLen]=val;
      asterisc_partial[stCurrentLen+1]='\0';
      asterisc_partial[stCurrentLen-1]='*';
    }
    else
    {
      asterisc_partial[stCurrentLen]=val;
      asterisc_partial[stCurrentLen+1]='\0';
    }
    stCurrentLen++;
    myGLCD.setColor(0, 0, 0);
    myGLCD.setBackColor (255, 255, 255);
    //myGLCD.print("________", CENTER, 20);
    myGLCD.print("_______________", CENTER, 290); //FIX
    //myGLCD.print(asterisc_partial, 56, 20);
    myGLCD.print(asterisc_partial, 120, 290); //set muncul angka pwd //FIX
  }
}
// Draw a red frame while a button is touched
void waitForIt(int x1, int y1, int x2, int y2)
{
  myGLCD.setColor(255, 0, 0);
  myGLCD.drawRoundRect (x1, y1, x2, y2);
  while (myTouch.dataAvailable())
    myTouch.read();
  myGLCD.setColor(255, 255, 255);
  myGLCD.drawRoundRect (x1, y1, x2, y2);
}
/*************************
**  Required functions  **
*************************/
void setup()
{
  pinMode(SW_pin, INPUT);
  digitalWrite(SW_pin, HIGH);
  pinMode(kiri1, OUTPUT);
  pinMode(kanan1, OUTPUT);
  pinMode(LR, OUTPUT);
  pinMode(RR, OUTPUT);
  pinMode(gigimundur, OUTPUT);
  pinMode(power, OUTPUT);
  Serial.begin(115200);
  {
  randomSeed(analogRead(0));
// Setup the LCD
  myGLCD.InitLCD();
   myGLCD.clrScr();
  myGLCD.setFont(BigFont);
  startup();
  delay(15000);
}
{
 myGLCD.InitLCD();
 myGLCD.clrScr();
 myGLCD.setFont(BigFont);
 for (int i=0; i<=100; i++){
 myGLCD.setColor(15,55+(2*i),0);
 myGLCD.print("Loading...", 100, 160, 0);
 myGLCD.setFont(SevenSegNumFont);
 myGLCD.printNumI(i,270,127,0);
 myGLCD.setFont(BigFont);
  if(i<=99){myGLCD.print("%", 345, 160, 0);}
  if(i>99){myGLCD.print("%", 375, 160, 0);}
}
  myGLCD.setColor(0, 0, 255);
  myGLCD.print("KURSI RODA ELEKTRIK", 0, 06, 0);
  myGLCD.setColor(0, 255, 0);
  myGLCD.print("by M.Adrian Saputra", 479, 0, 90);
  myGLCD.setColor(2550, 0, 0);
  myGLCD.print("POLITEKNIK NEGERI SRIWIJAYA", 50, 300, 0);
  myGLCD.setColor(255, 0, 255);
  myGLCD.print("M.MUKHLIS D.P", 0, 319, 270);
/*
  myGLCD.setFont(SevenSegNumFont);
  myGLCD.setColor(0, 255, 0);
  myGLCD.print("45", 65, 75, 45);
  myGLCD.print("90", 350, 50, 90);
  myGLCD.print("180", 420, 250, 180);
*/
delay(2000);
}
// Initial setup
  myGLCD.InitLCD();
  myGLCD.clrScr();
  myTouch.InitTouch();
  myTouch.setPrecision(PREC_MEDIUM);
  myTouch.InitTouch(TOUCH_ORIENTATION);
  dispx=myGLCD.getDisplayXSize();
  dispy=myGLCD.getDisplayYSize();
  text_y_center=(dispy/2)-6;
  myGLCD.setFont(BigFont);
  myGLCD.setBackColor(0, 0, 255);
  drawButtons();
}
void startup()
{
  myGLCD.setColor(255, 0, 0);
  myGLCD.fillRect(0, 0, dispx-1, 13);
  myGLCD.setColor(255, 255, 255);
  myGLCD.setBackColor(255, 0, 0);
  myGLCD.drawLine(0, 14, dispx-1, 14);
  myGLCD.print("KURSI RODA ELEKTRIK", CENTER, 1);
  myGLCD.setBackColor(0, 0, 0);
  myGLCD.print("INSTRUKSI", CENTER, 30);
  myGLCD.print("Masukkan Password", LEFT, 76);
  myGLCD.print("Kemudian Tekan Tombol Enter", LEFT, 94);
  myGLCD.print("Password Benar Joystick Fungsi", LEFT, 112);
  myGLCD.print("Tekan Password Lagi Untuk", LEFT, 130);
  myGLCD.print("Mematikan Joystick", LEFT, 148);
  myGLCD.print("JANGAN LUPA PASSWORD", CENTER, 170);
  myGLCD.print("KEAMANAN NO.1 !!!", CENTER, 187);
  myGLCD.print("INGAT JANGAN SAMPAI SALAH", CENTER, 226);
  }
void loop()
{
 while (true)
 {
  powerState = digitalRead(power);
 if (analogRead(X_pin)<600 && analogRead(X_pin)>400 && analogRead(Y_pin)>400 &&
analogRead(Y_pin)<600 && (powerState == HIGH))
 {
  Serial.println("\n\nnetral");
  digitalWrite(LR, HIGH);
  digitalWrite(RR, HIGH);
  analogWrite(kanan1, 0);
  analogWrite(kiri1, 0);
  delay(50);}
  if (analogRead(X_pin)<401 && analogRead(Y_pin)>400 && analogRead(Y_pin)<660 &&
(powerState == HIGH)){
  Serial.println("\n\nmaju");
  analogWrite(kanan1, 120);
  analogWrite(kiri1, 120);
  Serial.println(nilai);
  digitalWrite(LR, HIGH);
  digitalWrite(RR, HIGH);
     }
  if (analogRead(Y_pin)>600 && analogRead(X_pin)>400 && analogRead(X_pin)<660 &&
(powerState == HIGH)){
  Serial.println("\n\nkiri");
analogWrite(kanan1, 120);
  analogWrite(kiri1, 120);
  digitalWrite(LR, HIGH);
  digitalWrite(RR, LOW);

  }
 if (analogRead(Y_pin)<401 && analogRead(X_pin)>400 && analogRead(X_pin)<660 &&
(powerState == HIGH)){
  Serial.println("\n\nkanan");
  analogWrite(kanan1, 120);
  analogWrite(kiri1,120);
  digitalWrite(RR, HIGH);
  digitalWrite(LR, LOW);
  }
if (analogRead(X_pin)>600 && analogRead(Y_pin)>400 && analogRead(Y_pin)<660 &&
(powerState == HIGH)){
  Serial.println("\n\nmundur");
analogWrite(kanan1, 120);
  analogWrite(kiri1, 120);
  Serial.println(nilai);
  digitalWrite(LR, LOW);
  digitalWrite(RR, LOW);
  }
 if (digitalRead(SW_pin)<1){
  Serial.println("\n\nnetral");
  digitalWrite(gigimundur, LOW);
  analogWrite(kanan1, 0);
  analogWrite(kiri1, 0);
  digitalWrite(LR, HIGH);
  digitalWrite(RR, HIGH);
  }
  if (myTouch.dataAvailable())
  {
   myTouch.read();
   x=myTouch.getX();
   y=myTouch.getY();
   if ((y>=0) && (y<=7))  // Upper row
   {
   if ((x>=0) && (x<=30))  // Button: 1
    {
    waitForIt(0, 0, 60, 60);
    updateStr('1');
    Serial.print("data=1"); // print as an ASCII-encoded decimal - same as "DEC"
    Serial.print("\t");    // prints a tab
    Serial.print("DEC=");
    Serial.print(49, DEC);  // print as an ASCII-encoded decimal
    Serial.print("\t");    // prints a tab
    Serial.print("HEX=");
    Serial.print(49, HEX);  // print as an ASCII-encoded hexadecimal
    Serial.print("\t");    // prints a tab
    Serial.print("OCT=");
    Serial.print(49, OCT);  // print as an ASCII-encoded octal
    Serial.print("\t");    // prints a tab
    Serial.print("BIN=");
    Serial.println(49, BIN);  // print as an ASCII-encoded binary
    }
    if ((x>=46) && (x<=75))  // Button: 2
    {
    waitForIt(100, 0, 160, 60);
    updateStr('2');
    Serial.print("data=2"); // print as an ASCII-encoded decimal - same as "DEC"
    Serial.print("\t");    // prints a tab
    Serial.print("DEC=");
    Serial.print(50, DEC);  // print as an ASCII-encoded decimal
    Serial.print("\t");    // prints a tab
    Serial.print("HEX=");
    Serial.print(50, HEX);  // print as an ASCII-encoded hexadecimal
    Serial.print("\t");    // prints a tab
    Serial.print("OCT=");
    Serial.print(50, OCT);  // print as an ASCII-encoded octal
    Serial.print("\t");    // prints a tab
    Serial.print("BIN=");
```

```
Serial.println(50, BIN);  // print as an ASCII-encoded binary
}
if ((x>=76) && (x<=110))  // Button: 3
{
waitForIt(200, 0, 260, 60);
updateStr('3');
Serial.print("data=3"); // print as an ASCII-encoded decimal - same as "DEC"
Serial.print("\t");    // prints a tab
Serial.print("DEC=");
Serial.print(51, DEC);  // print as an ASCII-encoded decimal
Serial.print("\t");    // prints a tab
Serial.print("HEX=");
Serial.print(51, HEX);  // print as an ASCII-encoded hexadecimal
Serial.print("\t");    // prints a tab
Serial.print("OCT=");
Serial.print(51, OCT);  // print as an ASCII-encoded octal
Serial.print("\t");    // prints a tab
Serial.print("BIN=");
Serial.println(51, BIN);  // print as an ASCII-encoded binary
}
if ((x>=111) && (x<=149))  // Button: 4
{
waitForIt(300, 0, 360, 60);
updateStr('4');
Serial.print("data=4"); // print as an ASCII-encoded decimal - same as "DEC"
Serial.print("\t");    // prints a tab
Serial.print("DEC=");
Serial.print(52, DEC);  // print as an ASCII-encoded decimal
Serial.print("\t");    // prints a tab
Serial.print("HEX=");
Serial.print(52, HEX);  // print as an ASCII-encoded hexadecimal
Serial.print("\t");    // prints a tab
Serial.print("OCT=");
Serial.print(52, OCT);  // print as an ASCII-encoded octal
Serial.print("\t");    // prints a tab
Serial.print("BIN=");
Serial.println(52, BIN);  // print as an ASCII-encoded binary
}
if ((x>=150) && (x<=180))  // Button: 5
{
waitForIt(400, 0, 460, 60);
updateStr('5');
Serial.print("data=5"); // print as an ASCII-encoded decimal - same as "DEC"
Serial.print("\t");    // prints a tab
Serial.print("DEC=");
Serial.print(53, DEC);  // print as an ASCII-encoded decimal
Serial.print("\t");    // prints a tab
Serial.print("HEX=");
Serial.print(53, HEX);  // print as an ASCII-encoded hexadecimal
Serial.print("\t");    // prints a tab
Serial.print("OCT=");
Serial.print(53, OCT);  // print as an ASCII-encoded octal
Serial.print("\t");    // prints a tab
Serial.print("BIN=");
Serial.println(53, BIN);  // print as an ASCII-encoded binary
}
}
if ((y>=8) && (y<=15))  // Center row
{
if ((x>=0) && (x<30))  // Button: 6
{
waitForIt(0, 80, 60, 140);
updateStr('6');
Serial.print("data=6"); // print as an ASCII-encoded decimal - same as "DEC"
Serial.print("\t");    // prints a tab
Serial.print("DEC=");
Serial.print(54, DEC);  // print as an ASCII-encoded decimal
Serial.print("\t");    // prints a tab
Serial.print("HEX=");
Serial.print(54, HEX);  // print as an ASCII-encoded hexadecimal
Serial.print("\t");    // prints a tab
Serial.print("OCT=");
Serial.print(54, OCT);  // print as an ASCII-encoded octal
Serial.print("\t");    // prints a tab
Serial.print("BIN=");
Serial.println(54, BIN);  // print as an ASCII-encoded binary
}
if ((x>=46) && (x<=75))  // Button: 7
{
waitForIt(100, 80, 160, 140);
updateStr('7');
Serial.print("data=7"); // print as an ASCII-encoded decimal - same as "DEC"
Serial.print("\t");    // prints a tab
Serial.print("DEC=");
Serial.print(55, DEC);  // print as an ASCII-encoded decimal
Serial.print("\t");    // prints a tab
Serial.print("HEX=");
Serial.print(55, HEX);  // print as an ASCII-encoded hexadecimal
Serial.print("\t");    // prints a tab
Serial.print("OCT=");
Serial.print(55, OCT);  // print as an ASCII-encoded octal
Serial.print("\t");    // prints a tab
Serial.print("BIN=");
Serial.println(55, BIN);  // print as an ASCII-encoded binary
}
if ((x>=76) && (x<=110))  // Button: 8
{
waitForIt(200, 80, 260, 140);
updateStr('8');
Serial.print("data=8"); // print as an ASCII-encoded decimal - same as "DEC"
Serial.print("\t");    // prints a tab
Serial.print("DEC=");
Serial.print(56, DEC);  // print as an ASCII-encoded decimal
Serial.print("\t");    // prints a tab
Serial.print("HEX=");
Serial.print(56, HEX);  // print as an ASCII-encoded hexadecimal
Serial.print("\t");    // prints a tab
Serial.print("OCT=");
Serial.print(56, OCT);  // print as an ASCII-encoded octal
Serial.print("\t");    // prints a tab
Serial.print("BIN=");
Serial.println(56, BIN);  // print as an ASCII-encoded binary
}
if ((x>=111) && (x<=149))  // Button: 9
{
waitForIt(300, 80, 360, 140);
updateStr('9');
Serial.print("data=9"); // print as an ASCII-encoded decimal - same as "DEC"
Serial.print("\t");    // prints a tab
Serial.print("DEC=");
Serial.print(57, DEC);  // print as an ASCII-encoded decimal
Serial.print("\t");    // prints a tab
Serial.print("HEX=");
Serial.print(57, HEX);  // print as an ASCII-encoded hexadecimal
Serial.print("\t");    // prints a tab
Serial.print("OCT=");
Serial.print(57, OCT);  // print as an ASCII-encoded octal
Serial.print("\t");    // prints a tab
Serial.print("BIN=");
Serial.println(57, BIN);  // print as an ASCII-encoded binary
}
if ((x>=110) && (x<=180))  // Button: 0
{
waitForIt(400, 80, 460, 140);
updateStr('0');
Serial.print("data=0"); // print as an ASCII-encoded decimal - same as "DEC"
Serial.print("\t");    // prints a tab
Serial.print("DEC=");
Serial.print(48, DEC);  // print as an ASCII-encoded decimal
Serial.print("\t");    // prints a tab
Serial.print("HEX=");
Serial.print(48, HEX);  // print as an ASCII-encoded hexadecimal
Serial.print("\t");    // prints a tab
Serial.print("OCT=");
Serial.print(48, OCT);  // print as an ASCII-encoded octal
Serial.print("\t");    // prints a tab
Serial.print("BIN=");
Serial.println(48, BIN);  // print as an ASCII-encoded binary
myGLCD.setColor(0, 0, 0);
myGLCD.setBackColor (255, 255, 255);
//myGLCD.print(asterisc, 56, 20);
 myGLCD.print(asterisc, 120, 290); //FIX piksel dan tampilin bintang kejora
 }
 }
 if ((y>=16) && (y<=120))  // Lower row
 {
```

```
if ((x>=0) && (x<=100))  // Button: CLEAR
{
waitForIt(0, 160,448, 240);
if(stCurrentLen>0)  //jika nilai lebih besar dari 0 atau ada inputan angka lebih dari 0
{
stCurrent[stCurrentLen-1]='\0'; //kosongkan
asterisc[stCurrentLen-1]='\0'; // kosongkan
asterisc_partial[stCurrentLen-1]='\0'; //kosongkan
stCurrentLen=stCurrentLen-1; //setiap di hapus kurangi 1
clearsmalldisplay(); //hapus lcd
myGLCD.setColor(0, 0, 0); //hitam
myGLCD.setBackColor (255, 255, 255);
myGLCD.print("_____", CENTER, 290); //FIX
//myGLCD.print(asterisc, 56, 20);
myGLCD.print(asterisc, 120, 290); //geser kanan angka pwd //FIX
}
else
{
stCurrent[0]='\0';
if (setoldcode_flag) //set kode lama
{
setoldcode_flag = 0;
setnewcode_flag = 0;
repeat_flag = 1;
cleardisplay();
blinkmsg("EXITING SETUP", "BLACK");
msgCurrent = "ENTER PASSCODE";
printmsg(msgCurrent, "BLACK");
}
else
{
if (setnewcode_flag) //kode baru
{
if (repeat_flag) //fix
{
setoldcode_flag = 0;
setnewcode_flag = 0;
repeat_flag = 1;
cleardisplay();
blinkmsg("EXITING SETUP", "BLACK");
msgCurrent = "ENTER PASSCODE";
printmsg(msgCurrent, "BLACK");
}
else
{
setoldcode_flag = 0;
setnewcode_flag = 1;
repeat_flag = 1;
cleardisplay();
msgCurrent = "NEW PASSCODE"; // wajib pakai kode yang baru
printmsg(msgCurrent, "BLACK");
}
```

```
 }
 else
 {
 cleardisplay();
 msgCurrent = "ENTER PASSCODE"; // wajib kode baru
 printmsg(msgCurrent, "BLACK");
 }
 }
 }
 }
 if ((x>=110) && (x<=180))  // Button: ENTER
 {
 waitForIt(450, 160, 460, 240);
 count++; // jika salah maka counter bertambah 1, jika sudah 3 nyalakan buzzer
 thisByte=atoi(stCurrent);//array to int
 stLastserial=thisByte+48;
 Serial.print("data="); // print as an ASCII-encoded decimal - same as "DEC"
 Serial.print(stCurrent);
 Serial.print("\t");    // prints a tab
 Serial.print("DEC=");
 Serial.println(stLastserial);  // print as an ASCII-encoded decimal
 Serial.print("\t");    // prints a tab
 Serial.print("HEX=");
 Serial.println(stLastserial, HEX);  // print as an ASCII-encoded hexadecimal
Serial.print("\t");    // prints a tab
Serial.print("OCT=");
Serial.println(stLastserial, OCT);  // print as an ASCII-encoded octal
Serial.print("\t");    // prints a tab
Serial.print("BIN=");
Serial.println(stLastserial, BIN);
 if (stCurrentLen==5) //batas 5 ikok pwd
 {
 stLast = stCurrent; //masukkan angka inputan kedalam memori pwd
 stCurrent[0]='\0';
 asterisc[0]='\0';
 asterisc_partial[0]='\0';
 stCurrentLen=0;
}
// print as an ASCII-encoded binary
if (stLast != "00000") //jika memori pwd tidak sama dengan 00000m maka wajib ganti
kode baru
{
if (setnewcode_flag) //jika sudah harus ganti maka wajib kita menggunakan pwb yang
baru
{
if (repeat_flag)
{
stTemp = stLast;//masukkan pwb baru kedalam memori pwd tetap tapi yang baru
repeat_flag = 0;
cleardisplay();
msgCurrent = "REPEAT PASSCODE";
printmsg(msgCurrent, "BLACK");
```

```
}
else
{
if (stLast == stTemp) //kita su8dah pakai kode baru
{
stCode = stLast;
setoldcode_flag = 0;
setnewcode_flag = 0;
repeat_flag = 1;
cleardisplay();
blinkmsg("CODE UPDATED!", "GREEN");
msgCurrent = "ENTER PASSCODE";
printmsg(msgCurrent, "BLACK");
}
else
{
repeat_flag = 1; //jika konfirmasi kode baru salah maka tampilkan
cleardisplay();
blinkmsg("CODE MISMATCH", "RED");
msgCurrent = "NEW PASSCODE";
printmsg(msgCurrent, "BLACK");
}
}
}
else
{
if (stLast == stCode)
{
if (setoldcode_flag)
{
setoldcode_flag = 0;
setnewcode_flag = 1;
cleardisplay();
msgCurrent = "NEW PASSCODE"; // masukkan kode baru ataupun lama
printmsg(msgCurrent, "BLACK");
}
else
{
if (lock_flag)
{
cleardisplay();
blinkmsg("CONTROL LOCKED", "GREEN"); //
lock_flag = 0;
analogWrite(kanan1, 0);
analogWrite(kiri1, 0);
digitalWrite(LR, HIGH);
digitalWrite(RR, HIGH);
digitalWrite(power, LOW);
}
else
{
cleardisplay();
blinkmsg("True Passcode", "GREEN"); //buka gerbang
lock_flag = 1;
count = 0;
digitalWrite(power, HIGH);
}
cleardisplay();
msgCurrent = "ENTER PASSCODE"; //permintaan masukkan kode lagi setelah
masukkan kode
printmsg(msgCurrent, "BLACK");
}
}
else
{
cleardisplay();
if (lock_flag)
blinkmsg("ACCESS DENIED", "RED");
else
blinkmsg("WRONG PASSCODE", "RED"); //salah pwd
printmsg(msgCurrent, "BLACK");
}
}
}
else
{
setoldcode_flag = 1; //blok ngisi pwd baru dari yang lama
setnewcode_flag = 0;
repeat_flag = 1;
cleardisplay();
blinkmsg("ENTERING SETUP", "BLACK");
msgCurrent = "OLD PASSCODE";
printmsg(msgCurrent, "BLACK");
}
}
else
{
cleardisplay();
blinkmsg("TOO SHORT", "RED"); //jangan pendek coyyy
stCurrent[0]='\0';
asterisc[0]='\0';
asterisc_partial[0]='\0';
stCurrentLen=0;
cleardisplay();
printmsg(msgCurrent, "BLACK");
}
}
}
}
// read the value from the sensor:
sensorValue = analogRead(sensorPin);  //masukkan nilai adc ke serail
//Serial.println(sensorValue);   // print ADC value of analog reading
}
```