LAMPIRAN

A. CODING ESP 32 CAM

```
/*********************************************************************
**********

 *  TITLE: Motion Sensor Security Camera using ESP32-CAM & Blynk

 *  Click on the following links to learn more.

 *  YouTube Video: https://youtu.be/LqX9EMFSoDA

 *  Related Blog : https://easyelectronicsproject.com/esp32-projects/

 *  by Tech StudyCell

*********************************************************************
**********/



/*********************************************************************
**********

 *  Preferences--> Aditional boards Manager URLs :
https://dl.espressif.com/dl/package_esp32_index.json,
http://arduino.esp8266.com/stable/package_esp8266com_index.json

 *  Board Settings:

 *  Board: "ESP32 Wrover Module"

 *  Upload Speed: "921600"

 *  Flash Frequency: "80MHz"

 *  Flash Mode: "QIO"

 *  Partition Scheme: "Hue APP (3MB No OTA/1MB SPIFFS)"

 *  Core Debug Level: "None"

 *  COM Port: Depends *On Your System*

 *
```

```
 *  GPIO 0 must be connected to GND pin while uploading the sketch

 *  After connecting GPIO 0 to GND pin, press the ESP32 CAM on-board RESET button to
put the board in flashing mode

**********************************************************************
***************/


#include "esp_camera.h"

#include <WiFi.h>

#include <WiFiClient.h>

#include <BlynkSimpleEsp32.h>

#define CAMERA_MODEL_AI_THINKER // Has PSRAM

#include "camera_pins.h"

#define BLYNK_PRINT Serial


#define PIR 2

#define PHOTO 14

#define LED 4

int trigPin = 12 ;

int echoPin = 13 ;

float duration_us, jarak;

BlynkTimer timer;


const char* ssid = "Sharfina";

const char* password = "12345678";

char auth[] = "ESfVqYHDbT6q8BMG_ilq96Q3P14v6QHw";  //sent by Blynk
```

```
String local_IP;

void startCameraServer();

void takePhoto()
{
  digitalWrite(LED, HIGH);
  delay(200);
  uint32_t randomNum = random(50000);
  Serial.println("http://"+local_IP+"/capture?_cb="+ (String)randomNum);
  Blynk.setProperty(V1, "urls", "http://"+local_IP+"/capture?_cb="+(String)randomNum);
  digitalWrite(LED, LOW);
  delay(1000);
}

void setup() {
  Serial.begin(115200);
  pinMode(LED,OUTPUT);       // Sebagai INDIKATOR
  pinMode(PIR,INPUT);        // Sensor PIR sebagai INPUT
  pinMode(trigPin, OUTPUT);   // Ultasonik Trigger sebagai OUTPUT Gelombang
  pinMode(echoPin, INPUT);    // Ultrasonik Echo sebagai INPUT Gelombang
  timer.setInterval(1000L, jarakRacun);

  Serial.setDebugOutput(true);
  Serial.println();
```

```
camera_config_t config;
config.ledc_channel = LEDC_CHANNEL_0;
config.ledc_timer = LEDC_TIMER_0;
config.pin_d0 = Y2_GPIO_NUM;
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;

// if PSRAM IC present, init with UXGA resolution and higher JPEG quality
//                      for larger pre-allocated frame buffer.
if(psramFound()){
```

```
  config.frame_size = FRAMESIZE_UXGA;
  config.jpeg_quality = 10;
  config.fb_count = 2;
} else {
  config.frame_size = FRAMESIZE_SVGA;
  config.jpeg_quality = 12;
  config.fb_count = 1;
}

// camera init
esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK) {
  Serial.printf("Camera init failed with error 0x%x", err);
  return;
}

sensor_t * s = esp_camera_sensor_get();
// initial sensors are flipped vertically and colors are a bit saturated
if (s->id.PID == OV3660_PID) {
  s->set_vflip(s, 1); // flip it back
  s->set_brightness(s, 1); // up the brightness just a bit
  s->set_saturation(s, -2); // lower the saturation
}
// drop down frame size for higher initial frame rate
s->set_framesize(s, FRAMESIZE_QVGA);
```

```
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  local_IP = WiFi.localIP().toString();
  Serial.println("' to connect");
  Blynk.begin(auth, ssid, password);
}

void jarakRacun(){
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration_us = pulseIn(echoPin, HIGH);
  jarak =  0.017 *duration_us ;
  Serial.print("Distance : ");
```

```
  Serial.print(jarak);
  Serial.print("CM");
  Serial.println("");
  Blynk.virtualWrite(V0, jarak);
  if(jarak >=13 ){

    Blynk.notify("Kurang Racun Tambah kan Segera !!");
    Serial.print("Kurang Racun");
    Serial.println("");
    }
  else{
  }
  delay(500);
}



void loop() {
  // put your main code here, to run repeatedly:
  Blynk.run();
  timer.run();
  jarakRacun();
  int sensorval= digitalRead(PIR);

  if(sensorval == HIGH){
  Serial.println("Send Notification");
```

```
  Blynk.notify("Ada Tikus");
  takePhoto();
  delay(10000);
  }
  if(digitalRead(PHOTO) == HIGH){
  Serial.println("Capture Photo");
  takePhoto();
  }
}
```

B. CODING ARDUINO

```
#define MakanPertama          DateTime(0, 1, 1,  23, 0, 0, 0) // jam 11


#define ResetEspa             DateTime(0, 1, 1,  22, 20, 0, 0)
#define ResetEspb             DateTime(0, 1, 1,  22, 40, 0, 0)
#define ResetEspc             DateTime(0, 1, 1,  23, 0, 0, 0)
#define ResetEspd             DateTime(0, 1, 1,  23, 20, 0, 0)
#define ResetEspe             DateTime(0, 1, 1,  23, 40, 0, 0)
#define ResetEspf             DateTime(0, 1, 1,  0, 0, 0, 0)
#define ResetEspg             DateTime(0, 1, 1,  0, 20, 0, 0)
#define ResetEsph             DateTime(0, 1, 1,  0, 40, 0, 0)
#define ResetEspi             DateTime(0, 1, 1,  1, 0, 0, 0)
#define ResetEspj             DateTime(0, 1, 1,  1, 20, 0, 0)
#define ResetEspk             DateTime(0, 1, 1,  1, 40, 0, 0)
#define ResetEspl             DateTime(0, 1, 1,  2, 0, 0, 0)
#define ResetEspm             DateTime(0, 1, 1,  2, 20, 0, 0)
#define ResetEspn             DateTime(0, 1, 1,  2, 40, 0, 0)
#define ResetEspo             DateTime(0, 1, 1,  3, 0, 0, 0)
#define ResetEspp             DateTime(0, 1, 1,  3, 20, 0, 0)
#define ResetEspq             DateTime(0, 1, 1,  3, 40, 0, 0)
#define ResetEspr             DateTime(0, 1, 1,  4, 0, 0, 0)
#define ResetEsps             DateTime(0, 1, 1,  4, 20, 0, 0)
#define ResetEspt             DateTime(0, 1, 1,  4, 40, 0, 0)
#define ResetEspu             DateTime(0, 1, 1,  5, 0, 0, 0)
```

```
#define pinservo                2
#define waktuservo_ON           1000//milidetik
#define servo_ON                65//derajat
#define servo_OFF               90//derajat

#define relay                   7

#include <Servo.h>
int pinIr =                     A0;
int ir    =                     LOW;
#define pinkiri                 3
#define pinkanan                4

Servo gerakkiri;
Servo gerakkanan;
#define gerakkiri_ON            90
#define gerakkanan_ON           90
#define gerakkiri_OFF           10
#define gerakkanan_OFF          0

#include <Wire.h>
#include "RTC_DS3231_RP.h"


Servo iniservo;
```

```
char weekDay[][7] = {"Minggu", "Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu" };
byte detikSebelumnya;
char buf[17];

void setup() {
  Serial.begin(9600);
  iniservo.attach(pinservo);
  iniservo.write(servo_OFF);

  pinMode(relay, OUTPUT);
  digitalWrite (relay, HIGH);
  pinMode(pinIr, INPUT);
  gerakkiri.attach(pinkiri);
  gerakkanan.attach(pinkanan);
  gerakkiri.write(gerakkiri_OFF);
  gerakkanan.write(gerakkanan_OFF);

  Wire.begin();
  rtc.begin();

  Serial.println("Sistem mulai");
  sprintf(buf, "Set waktu 1 = %02d:%02d (%lu)", MakanPertama.hour(),
MakanPertama.minute(), MakanPertama.get());
  Serial.println(buf);

}
```

```
void loop(){
 ir = digitalRead(pinIr);
 DateTime now = rtc.now();
 Serial.println(weekDay[now.dayOfWeek()]);//untuk tampilan hari

 Serial.print(now.date(), DEC); //untuk tampilan tanggal
 Serial.print("/");
 Serial.print(now.month(), DEC);//untuk tampilan bulan
 Serial.print("/");
 Serial.println(now.year(), DEC);//untuk tampilan tahun
 if(ir == HIGH){

 gerakkiri.write(gerakkiri_OFF);
 gerakkanan.write(gerakkanan_OFF);

}else {

 gerakkiri.write(gerakkiri_ON);
 gerakkanan.write(gerakkanan_ON);
 delay(2000);
}
 if (detikSebelumnya != now.second())
 {
   sprintf(buf, "%02d:%02d:%02d", now.hour(), now.minute(), now.second());
   Serial.print(buf);
```

```
  detikSebelumnya = now.second();
  uint32_t epoch = now.get() % 86400;//hanya jam menit detik

  if (epoch == MakanPertama.get())

  {
    char buf[17];
    sprintf(buf, "SEKARANG: %02d:%02d", now.hour(), now.minute());//untuk tampilan saat waktu makan

    Serial.print(buf);

    Serial.print("Waktunya Makan!!");//untuk tampilan saat waktu makan
    Serial.println(buf);

    //tambah atau kurangi program ini untuk setting berapa kali servo berjalan
    //di setting 2X
    iniservo.write(servo_ON);
    delay(1000);
    iniservo.write(servo_OFF);
    delay(1000);
  }
  if (epoch == ResetEspa.get())
  { Serial.println("Reset berhasil 1");
    char buf[17];
    sprintf(buf, now.minute());
```

```
  digitalWrite(relay,LOW);
  delay(500);
  digitalWrite(relay,HIGH);
}
if (epoch == ResetEspb.get())
{ Serial.println("Reset berhasil 2");
  char buf[17];
  sprintf(buf, now.minute());
  digitalWrite(relay,LOW);
  delay(500);
  digitalWrite(relay,HIGH);
}if (epoch == ResetEspc.get())
{ Serial.println("Reset berhasil 3");
  char buf[17];
  sprintf(buf, now.minute());
  digitalWrite(relay,LOW);
  delay(500);
  digitalWrite(relay,HIGH);
}if (epoch == ResetEspd.get())
{ Serial.println("Reset berhasil 4");
  char buf[17];
  sprintf(buf, now.minute());
  digitalWrite(relay,LOW);
  delay(500);
  digitalWrite(relay,HIGH);
```

```
}if (epoch == ResetEspe.get())
{ Serial.println("Reset berhasil 5");
  char buf[17];
  sprintf(buf, now.minute());
  digitalWrite(relay,LOW);
  delay(500);
  digitalWrite(relay,HIGH);
}if (epoch == ResetEspf.get())
{ Serial.println("Reset berhasil 6");
  char buf[17];
  sprintf(buf, now.minute());
  digitalWrite(relay,LOW);
  delay(500);
  digitalWrite(relay,HIGH);
}if (epoch == ResetEspg.get())
{ Serial.println("Reset berhasil 7");
  char buf[17];
  sprintf(buf, now.minute());
  digitalWrite(relay,LOW);
  delay(500);
  digitalWrite(relay,HIGH);
}if (epoch == ResetEsph.get())
{ Serial.println("Reset berhasil 8");
  char buf[17];
  sprintf(buf, now.minute());
```

```
  digitalWrite(relay,LOW);
  delay(500);
  digitalWrite(relay,HIGH);
}if (epoch == ResetEspi.get())
{ Serial.println("Reset berhasil 9");
  char buf[17];
  sprintf(buf, now.minute());
  digitalWrite(relay,LOW);
  delay(500);
  digitalWrite(relay,HIGH);
}if (epoch == ResetEspj.get())
{ Serial.println("Reset berhasil 10");
  char buf[17];
  sprintf(buf, now.minute());
  digitalWrite(relay,LOW);
  delay(500);
  digitalWrite(relay,HIGH);
}if (epoch == ResetEspk.get())
{ Serial.println("Reset berhasil 11");
  char buf[17];
  sprintf(buf, now.minute());
  digitalWrite(relay,LOW);
  delay(500);
  digitalWrite(relay,HIGH);
}if (epoch == ResetEspl.get())
```

```
{ Serial.println("Reset berhasil 12 ");
  char buf[17];
  sprintf(buf, now.minute());
  digitalWrite(relay,LOW);
  delay(500);
  digitalWrite(relay,HIGH);
}if (epoch == ResetEspm.get())
{ Serial.println("Reset berhasil13");
  char buf[17];
  sprintf(buf, now.minute());
  digitalWrite(relay,LOW);
  delay(500);
  digitalWrite(relay,HIGH);
}if (epoch == ResetEspn.get())
{ Serial.println("Reset berhasil 14");
  char buf[17];
  sprintf(buf, now.minute());
  digitalWrite(relay,LOW);
  delay(500);
  digitalWrite(relay,HIGH);
}if (epoch == ResetEspo.get())
{ Serial.println("Reset berhasil 15");
  char buf[17];
  sprintf(buf, now.minute());
  digitalWrite(relay,LOW);
```

```
  delay(500);
  digitalWrite(relay,HIGH);
}if (epoch == ResetEspp.get())
{ Serial.println("Reset berhasil 16");
  char buf[17];
  sprintf(buf, now.minute());
  digitalWrite(relay,LOW);
  delay(500);
  digitalWrite(relay,HIGH);
}if (epoch == ResetEspq.get())
{ Serial.println("Reset berhasil 17");
  char buf[17];
  sprintf(buf, now.minute());
  digitalWrite(relay,LOW);
  delay(500);
  digitalWrite(relay,HIGH);
}if (epoch == ResetEspr.get())
{ Serial.println("Reset berhasil 18");
  char buf[17];
  sprintf(buf, now.minute());
  digitalWrite(relay,LOW);
  delay(500);
  digitalWrite(relay,HIGH);
}if (epoch == ResetEsps.get())
{ Serial.println("Reset berhasil 19");
```

```
    char buf[17];
    sprintf(buf, now.minute());
    digitalWrite(relay,LOW);
    delay(500);
    digitalWrite(relay,HIGH);
   }if (epoch == ResetEspt.get())
   { Serial.println("Reset berhasil 20");
    char buf[17];
    sprintf(buf, now.minute());
    digitalWrite(relay,LOW);
    delay(500);
    digitalWrite(relay,HIGH);
   }if (epoch == ResetEspu.get())
   { Serial.println("Reset berhasil 21");
    char buf[17];
    sprintf(buf, now.minute());
    digitalWrite(relay,LOW);
    delay(500);
    digitalWrite(relay,HIGH);

   }
  }
  delay(1000);
}
```

C. CODING ADJUST RTC

```
// Date and time functions using a RX8025 RTC connected via I2C and Wire lib

#include <Wire.h>
#include "RTC_DS3231_RP.h"

char weekDay[][7] = {"Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" };

//year, month, date, hour, min, sec and week-day(starts from 0 and goes to 6)
//writing any non-existent time-data may interfere with normal operation of the RTC.
//Take care of week-day also.
DateTime dt(2022, 7, 17, 02, 26, 0, 0);


void setup ()
{
   Serial.begin(9600);
   Wire.begin();
   rtc.begin();
   rtc.setDateTime(dt); //Adjust date-time as defined 'dt' above
}

void loop ()
{
```

```
    DateTime now = rtc.now(); //get the current date-time
    Serial.print(now.year(), DEC);
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.date(), DEC);
    Serial.print(' ');
    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.println();
    Serial.print(weekDay[now.dayOfWeek()]);
    Serial.println();
    delay(1000);
}
```

D. DATASHEET ESP 32 CAM

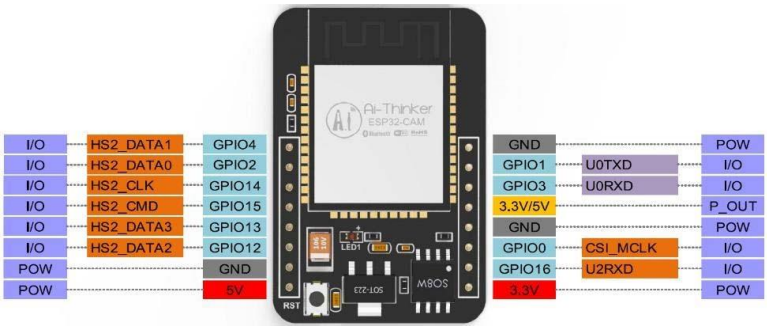
INTRODUCTION

ESP32-CAM is a low-cost ESP32-based development board with onboard camera, small in size. It is an ideal solution for IoT application, prototypes constructions and DIY projects.

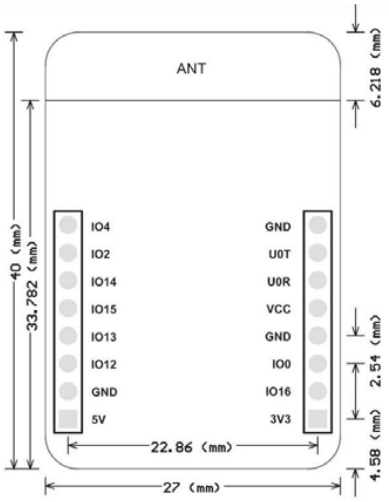
The board integrates WiFi, traditional Bluetooth and low power BLE , with 2 high- performance 32-bit LX6 CPUs. It adopts 7-stage pipeline architecture, on-chip sensor, Hall sensor, temperature sensor and so on, and its main frequency adjustment rangesfrom 80MHz to 240MHz.

Fully compliant with WiFi 802.11b/g/n/e/i and Bluetooth 4.2 standards, it can be used asa master mode to build an independent network controller, or as a slave to other host MCUs to add networking capabilities to existing devices

ESP32-CAM can be widely used in various IoT applications. It is suitable for home smart devices, industrial wireless control, wireless monitoring, QR wireless identification, wireless positioning system signals and other IoT applications. It is anideal solution for IoT applications.



SCHEMATIC DIAGRAM



DIMENSION DIAGRAM

Notes:

Please be sure that the power supply for the module should be at least 5V 2A, otherwise maybe there would be water ripple appearing on the image

1. ESP32 GPIO32 pin is used to control the power of the camera, so when the camera isin working, pull GPIO32 pin low.

2. Since IO pin is connected to camera XCLK, it should be left floating in using, and donot connect it to high/low level.

3. The product has been equipped with default firmware before leaving the factory, andwe do not provide additional ones for you to download. So, please be cautious when you choose to burn other firmwares.

FEATURES

- Up to 160MHz clock speed, Summary computing power up to 600 DMIPS
- Built-in 520 KB SRAM, external 4MPSRAM
- Supports UART/SPI/I2C/PWM/ADC/DAC
- Support OV2640 and OV7670 cameras, Built-in Flash lamp.
- Support image WiFI upload
- Support TF card
- Supports multiple sleep modes.
- Embedded Lwip and FreeRTOS
- Supports STA/AP/STA+AP operation mode
- Support Smart Config/AirKiss technology
- Support for serial port local and remote firmware upgrades (FOTA)

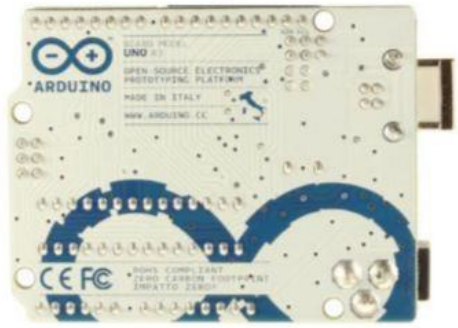
SPECIFICATION

- SPI Flash: default 32Mbit
- RAM: built-in 520 KB+external 4MPSRAM
- Dimension: 27*40.5*4.5（±0.2）mm/1.06*1.59*0.18”
- Bluetooth: Bluetooth 4.2 BR/EDR and BLE standards
- Wi-Fi: 802.11b/g/n/e/i
- Support Interface: UART, SPI, I2C, PWM
- Support TF card: maximum support 4G
- IO port: 9
- Serial Port Baud-rate: Default 115200 bps
- Image Output Format: JPEG(OV2640 support only), BMP, GRAYSCALE
- Spectrum Range: 2412 ~2484MHz
- Antenna: onboard PCB antenna, gain 2dBi
- Transmit Power: 802.11b: 17±2 dBm (@11Mbps);
  802.11g: 14±2 dBm (@54Mbps);
  802.11n: 13±2 dBm (@MCS7)
- receiving Sensitivity: CCK, 1 Mbps : -90dBm;
  CCK, 11 Mbps: -85dBm;
  54 Mbps (1/2 BPSK): -88dBm;
  54 Mbps (3/4 64-QAM): -70dBm; MCS7 (65 Mbps, 72.2 Mbps): -67dBm
- Power consumption: Turn off the flash: 180mA@5V
  Turn on the flash and adjust the brightness to the maximum: 310 Ma 5V
- Deep-sleep: the lowest power consumption can reach 6mA@5VModerm-sleep: up to 20mA@5V
- Light-sleep: up to 6.7mA@5V
- Security: WPA/WPA2/WPA2-Enterprise/WPS
- Power supply range: 5V
- Operating temperature: -20 °C ~ 85 °C
- Sorage environment: -40 °C ~ 90 °C, < 90%RH
- Weight: 10g

E. DATASHEET ARDUINO UNO

Arduino Uno

Arduino Uno R3 Front



Arduino Uno R3 Back

Arduino Uno R2 Front

Arduino Uno SMD

Arduino Uno Front

Arduino Uno Back

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

Schematic & Reference Design

EAGLE files: arduino-uno-Rev3-reference-design.zip (NOTE: works with Eagle 6.0 and newer)
Schematic: arduino-uno-Rev3-schematic.pdf
Note: The Arduino reference design can use an Atmega8, 168, or 328, Current models use an ATmega328, but an Atmega8 is shown in the schematic for reference. The pin configuration is identical on all three processors.

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 is it 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.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial 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 Uno's digital pins.
The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

Programming

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno from the **Tools > Board** menu (according to the microcontroller on your board). For details, see the reference and tutorials.
The ATmega328 on the Arduino Uno comes preburned with a bootloader that 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 these instructions for details.
The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available . The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

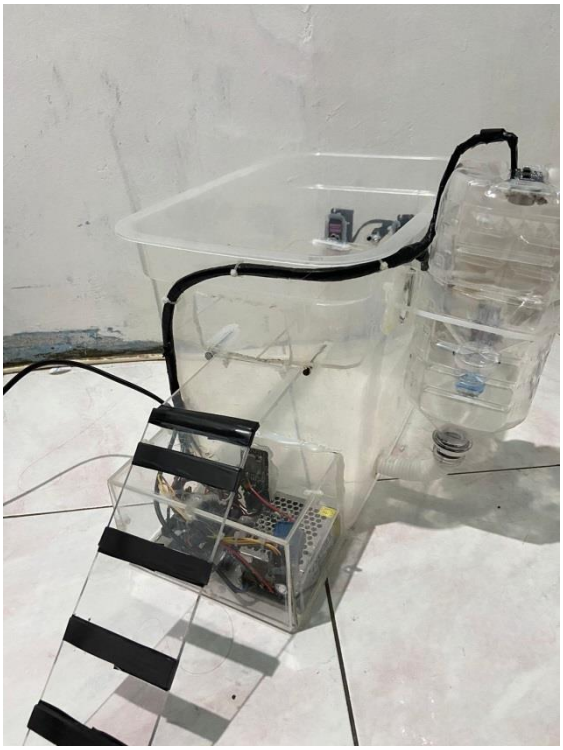
- On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2.
- On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

Automatic (Software) Reset

F. BENTUK MEKANIK TAMPAK DEPAN

G. BENTUK MEKANIK TAMPAK DALAM

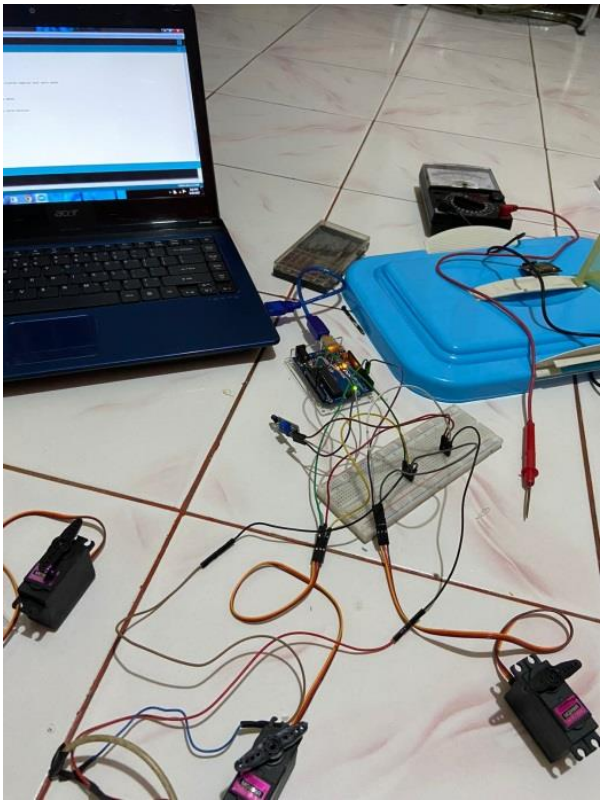


H. PENEMPATAN PERANGKAP TIKUS

I. TIKUS TERPERANGKAP

J. HASIL TIKUS TERPERANGKAP

K. MELAKUKAN CODING



L. PEMBUATAN PERANGKAP TIKUS