

ARDUINO

How to use / Notes

This document contains notes related to HW and SW projects based on Arduino hardware. Some of them are included from sources on the internet while others are created. The goal is to have a reference of the used boards and extension boards available, including references to software (libraries) and/or example code.

Table of contents

1 ARDUINO IDE 4
1.1 INSTALLATION 4
1.2 LOCATION OF THE CODE + WAY OF WORKING 4
1.3 SETUP BOARD AND PORT TO COMPILE AND UPLOAD SOFTWARE 4
1.3.1 Define the target board 4
1.3.2 Define the serial port 4
1.3.3 Test: compiling and uploading the code 5

2 ARDUINO BOARDS 6
2.1 ATMEGA 2560 6
2.2 NANO 8
2.3 ATMEGA 328P (NANO) PIN MAPPING 8
2.4 RF NANO 9
2.5 A* 328PB MICRO 10
2.6 WEMOS D1 MINI 12
2.7 TENSTAR ROBOT ESP32-C3 SUPER MINI 13
2.8 ESP32 DEVELOPMENT BOARD TYPE-C CP21023/CH340C WI FI + BLUETOOTH 16

3 LIBRARY INSTALLATIONS 17

4 EXTENSION BOARDS 18
4.1 ADAFRUIT PCA9685 16 CHANNEL SERVO DRIVER 18
4.2 L298N DUAL MOTOR CONTROLLER 19
4.3 WEMOS D1 MINI DSP8266 MATRIX LED SHIELD 8X8 20
4.4 HC SR-04 ULTRASONIC DISTANCE SENSOR 21
4.5 IR SENSOR MODULE 23
4.6 SERVO 23
4.7 WEMOS D1 MINI DSP8266 WIFI 25
4.8 FC16 MATRIX LCD (MAX7219) 26
4.9 RAMPS INTERFACE BOARD 29
4.10 A4988 STEPPER MOTOR DRIVER 31
4.11 A3967 STEPPER MOTOR DRIVER (EASYDRIVER) 32
4.12 STEPPER MOTOR PM15S-020 34
4.13 DRV8825 STEPPER MOTOR DRIVER 35
4.14 L78L33A VOLTAGE REGULATOR 3.3V 36
4.15 RAMPS FULL GRAPIC SMART CONTROLLER 37
4.15.1 LCD ST7920 controller, 128 x 64 pixels 37
4.15.2 Buzzer 39
4.15.3 Button 39
4.15.4 Rotary (rotation) 40
4.15.5 Rotary (push button) 41
4.16 RAMPS STEPPER MOTOR 42
4.17 RAMPS PINS CONNECTIONS TO ATMEGA2560 42
4.18 HC-05 BLUETOOTH SERIAL INTERFACE 43
4.18.1 Configuration for Master or Slave 43
4.19 TFT COLOR DISPLAY 320 X 480 ILI9486 44
4.20 TFT COLOR DISPLAY 320 X 480 SPI ILI9488 45
4.21 JOYSTICK SHIELD V1.A 46
4.22 WIRELESS MODULE NFR24L01 47
4.23 TEMPERATURE & HUMIDITY SENSOR DHT-22 AM2302 48
4.24 RAIN SENSOR 48
4.25 HC-SR501 PIR SENSOR 48
4.26 SR602 MICRO PIR SENSOR 49
4.27 MT3608 DC-DC BUCK CONVERTER/BOOSTER 2A 51
4.28 BQ25185 SOLAR CHARGER + 5V BOOSTER 52
4.29 RTC/EPROM MODULE DS1307/24C32 53
4.30 RTC/EEPROM MODULE DS3231/AT24C32 53
4.31 MICROSD CARD ADAPTER 54
4.32 VS1053 CODEC / MP3 PLAYER 55
4.33 YX6300 SERIAL CONTROLLED MP3 PLAYER 56
4.34 PAM8610 / HW-210 STEREO AUDIO AMPLFIER 2 X 10W CLASS-D 12V 57
4.35 LOW POWER MODE CIRCUIT (ON / OFF) 57
4.36 LEVEL SHIFTER CIRCUIT 1 58
4.37 LEVEL SHIFTER CIRCUIT 2 58

4.38 DY-SV19R SPEECH MODULE MP3/WAV 59
4.39 TF CARD MP3 AUDIO DECODER BOARD (2W, 5V) 65

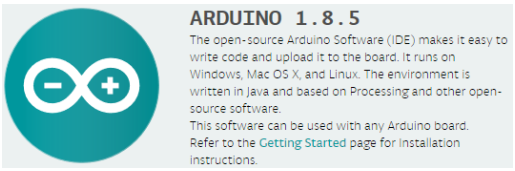
5 CODING NOTES 67
5.1 INSTALLATION OF LIBRARIES 67
5.1.1 Library installation from the GUI: Manage Libraries 67
5.1.2 Library installation: Manual (.zip file) 68
5.2 VARIABLE TYPES 68
5.3 COMMANDS 69
5.4 INTERRUPTS 69
5.4.1 Timer generating an interrupt every N milliseconds 69
5.4.2 Interrupt from an input pin 70
5.5 GRAPHICAL LIBRARIES 71
5.5.1 Adafruit GFX library 71
5.5.2 TFT eSPI 71

6 COLOR CODING GUIDES 72
6.1 INDUCTORS 72
6.2 RESISTORS 72

1 Arduino IDE

1.1 Installation

The Arduino IDE is downloaded from: https://www.arduino.cc/en/main/software
Version which is used: 1.8.5, updated, current (2026) used version is v2.3.6



1.2 Location of the code + way of working

By default, the Arduino IDE stores the projects in c:\Documents\Arduino\

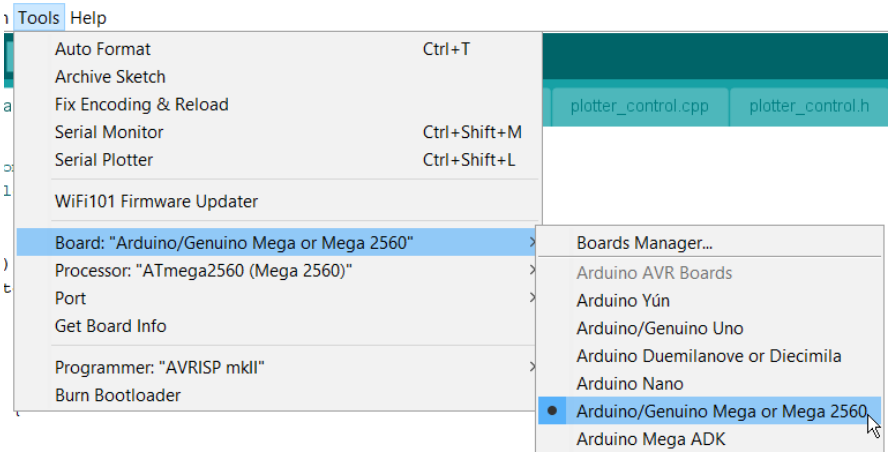
The (to be) developed code is not checked-in and checked-out by using a revision system, for now the way of working is to save a new project on a major new step in the development process. The Arduino IDE makes this very easy, since a 'Save as' saves all files of the project into the new project folder.
Method: Use 'Save as', move up to the projects folder, define a new name, which is the new project folder.

Differences in Arduino code related to other systems:
A project has two procedures, called setup() and loop(), after powering on the Arduino board, first the setup() is executed, then the loop() is started. These functions are called from the main() function.
For details see: {Arduino install}/hardware/arduino/avr/cores/arduino/main.cpp

1.3 Setup Board and Port to compile and upload software

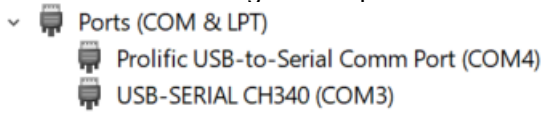
1.3.1 Define the target board

In Arduino IDE: Tools -> Board: -> Select the Mega2560 board.
Note: This is an IDE setting, thus not related to the project file (.ino), which is strange/inconvenient.... (?)



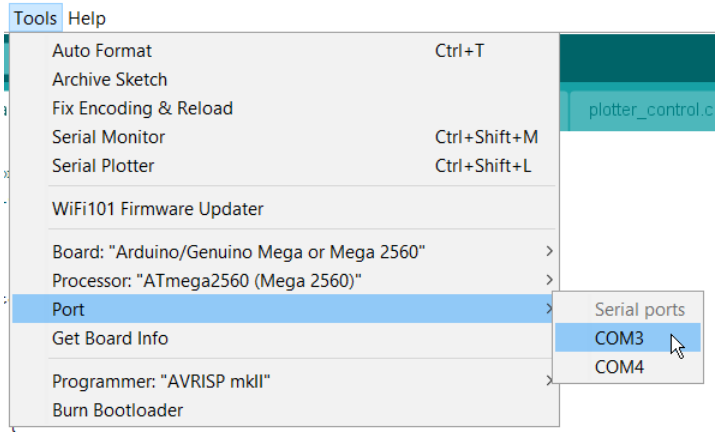
1.3.2 Define the serial port

Start the Device Manager and open the section Ports to find out which port is used.

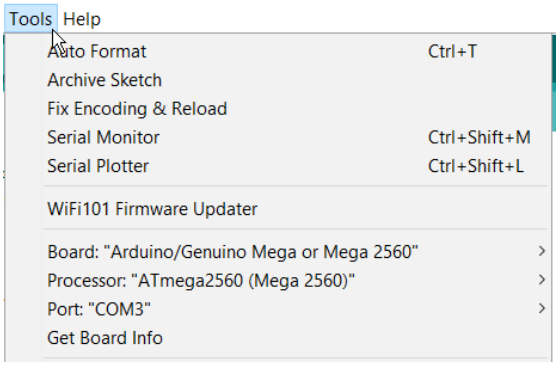


In this case COM3 is used (USB-SERAL CH340)

In Arduino IDE: Tools -> Port -> Select the correct COM port found in the Device Manager, (COM3)



After both definitions the tools menu directly shows these settings:

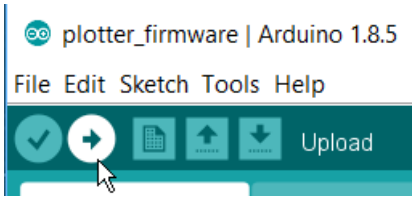


1.3.3 Test: compiling and uploading the code

If you want to test these settings then write a simple program. For instance:

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  while (1) {
      Serial.println("Ok");
  }
}
```

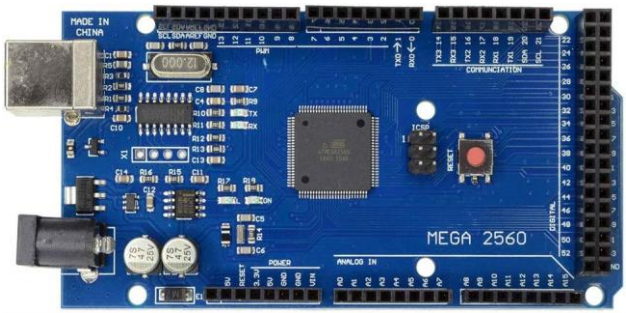
Press the Upload button which also compiles the code:
This is yet another strange thing: Each time this button is pressed the code will be compiled, a normal environment would know if the compiled code is out of date related to the source code and minimize compile time.



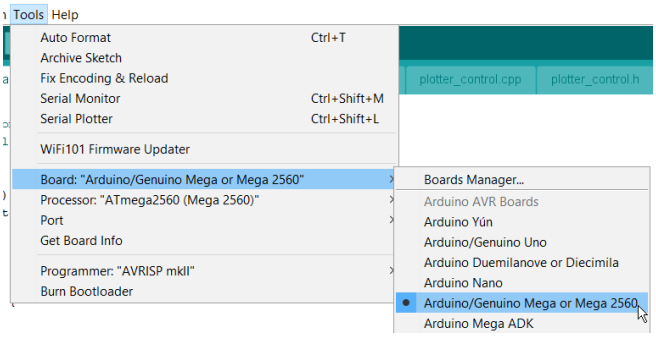
Then open the Serial Monitor, that should show lots of 'OK's. Tools -> Serial Monitor or keys: Ctrl + Shift + m.

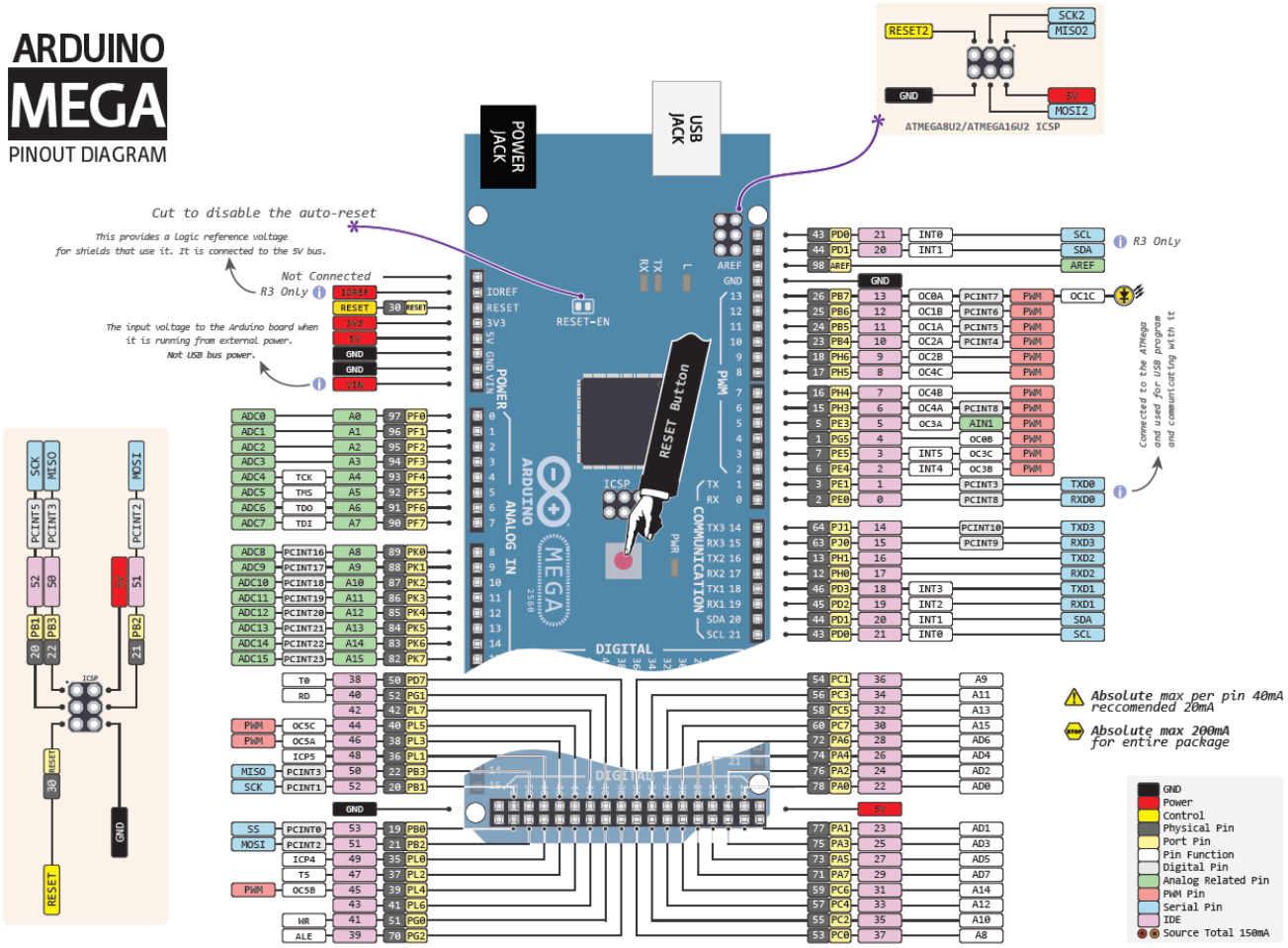
2 Arduino Boards

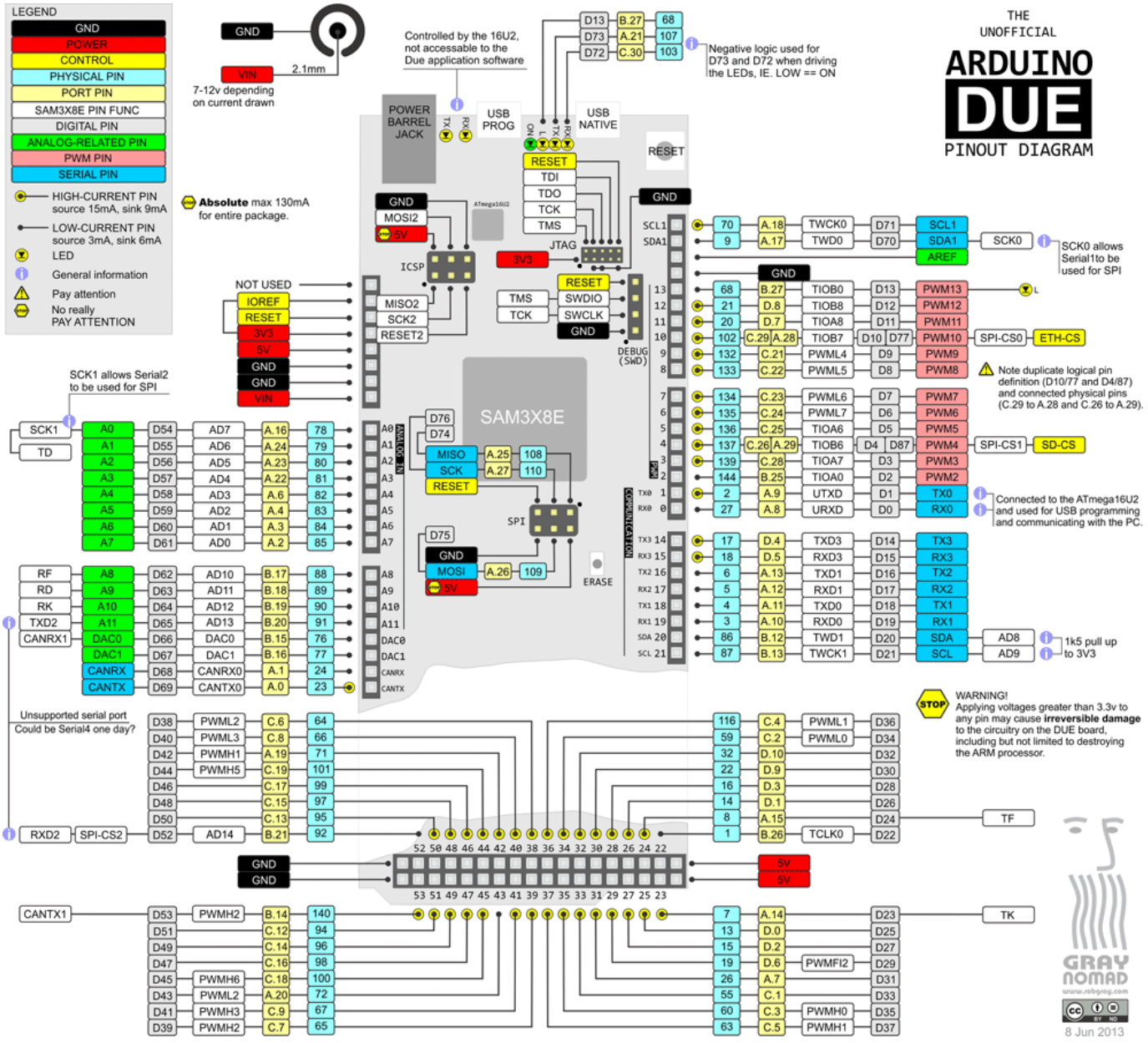
2.1 ATMEGA 2560



Select target board:
Arduino IDE: Tools -> Board: -> Select the Mega2560 board.



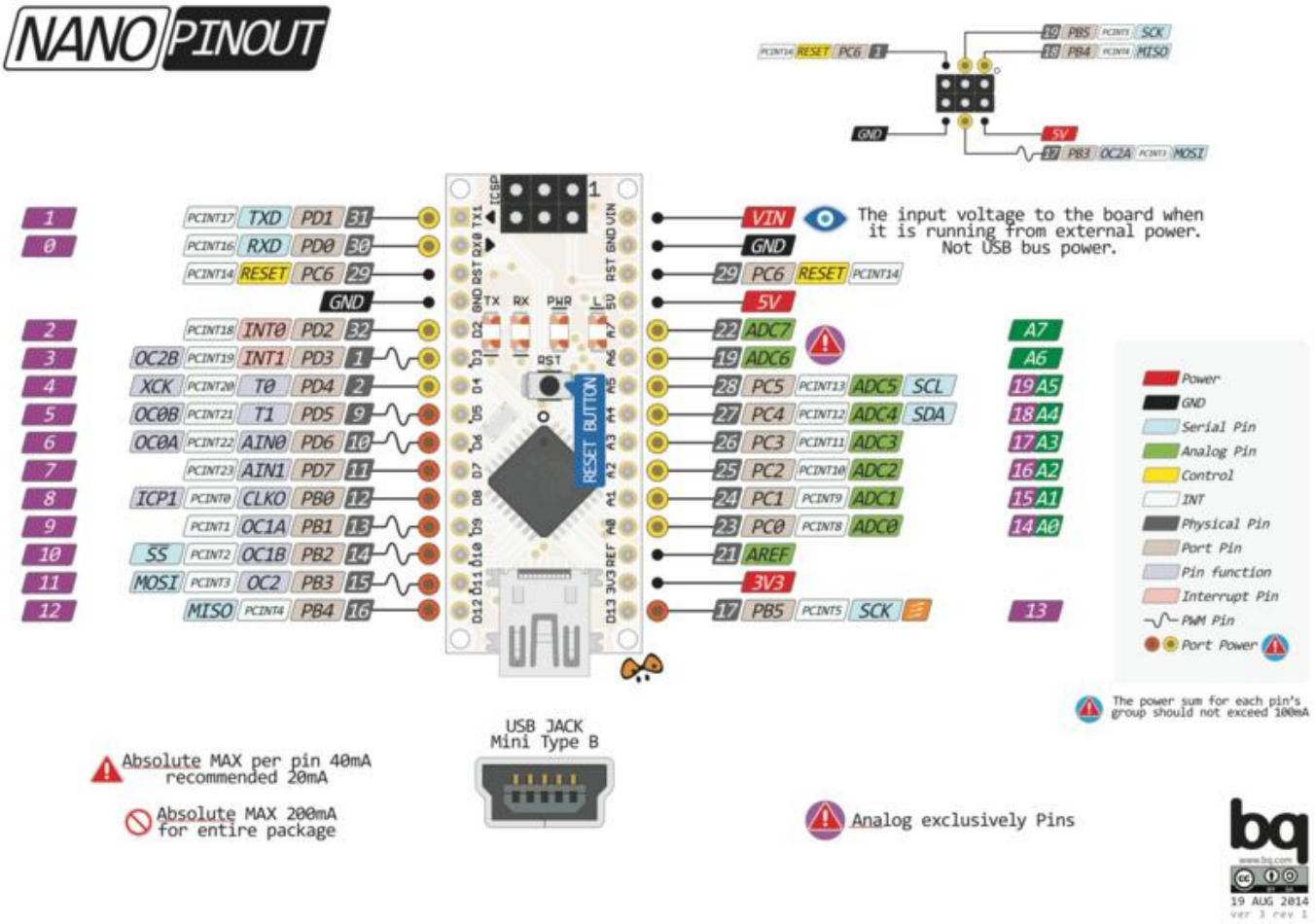




Note: To refer to digital pins use the number in the Gray part, like pinMode(**3**, OUTPUT);

2.2 Nano

Note: Downloading a program for the Nano board: Use Tools->Processor: **ATMEG328P (Old bootloader)**



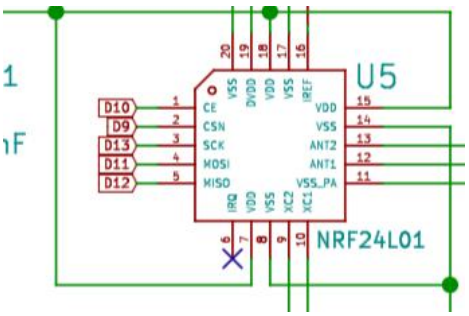
2.3 ATMEGA 328P (NANO) pin mapping

Arduino function				Arduino function
reset	(PCINT14/RESET) PC6	1	28 PC5 (ADC5/SCL/PCINT13)	analog input 5
digital pin 0 (RX)	(PCINT16/RXD) PD0	2	27 PC4 (ADC4/SDA/PCINT12)	analog input 4
digital pin 1 (TX)	(PCINT17/TXD) PD1	3	26 PC3 (ADC3/PCINT11)	analog input 3
digital pin 2	(PCINT18/INT0) PD2	4	25 PC2 (ADC2/PCINT10)	analog input 2
digital pin 3 (PWM)	(PCINT19/OC2B/INT1) PD3	5	24 PC1 (ADC1/PCINT9)	analog input 1
digital pin 4	(PCINT20/XCK/T0) PD4	6	23 PC0 (ADC0/PCINT8)	analog input 0
VCC	VCC	7	22 GND	GND
GND	GND	8	21 AREF	analog reference
crystal	(PCINT6/XTAL1/TOSC1) PB6	9	20 AVCC	VCC
crystal	(PCINT7/XTAL2/TOSC2) PB7	10	19 PB5 (SCK/PCINT5)	digital pin 13
digital pin 5 (PWM)	(PCINT21/OC0B/T1) PD5	11	18 PB4 (MISO/PCINT4)	digital pin 12
digital pin 6 (PWM)	(PCINT22/OC0A/AIN0) PD6	12	17 PB3 (MOSI/OC2A/PCINT3)	digital pin 11(PWM)
digital pin 7	(PCINT23/AIN1) PD7	13	16 PB2 (SS/OC1B/PCINT2)	digital pin 10 (PWM)
digital pin 8	(PCINT0/CLKO/ICP1) PB0	14	15 PB1 (OC1A/PCINT1)	digital pin 9 (PWM)

2.4 RF Nano

Note: Downloading a program for the RF Nano board: Use Tools->Processor: **ATMEG328P**

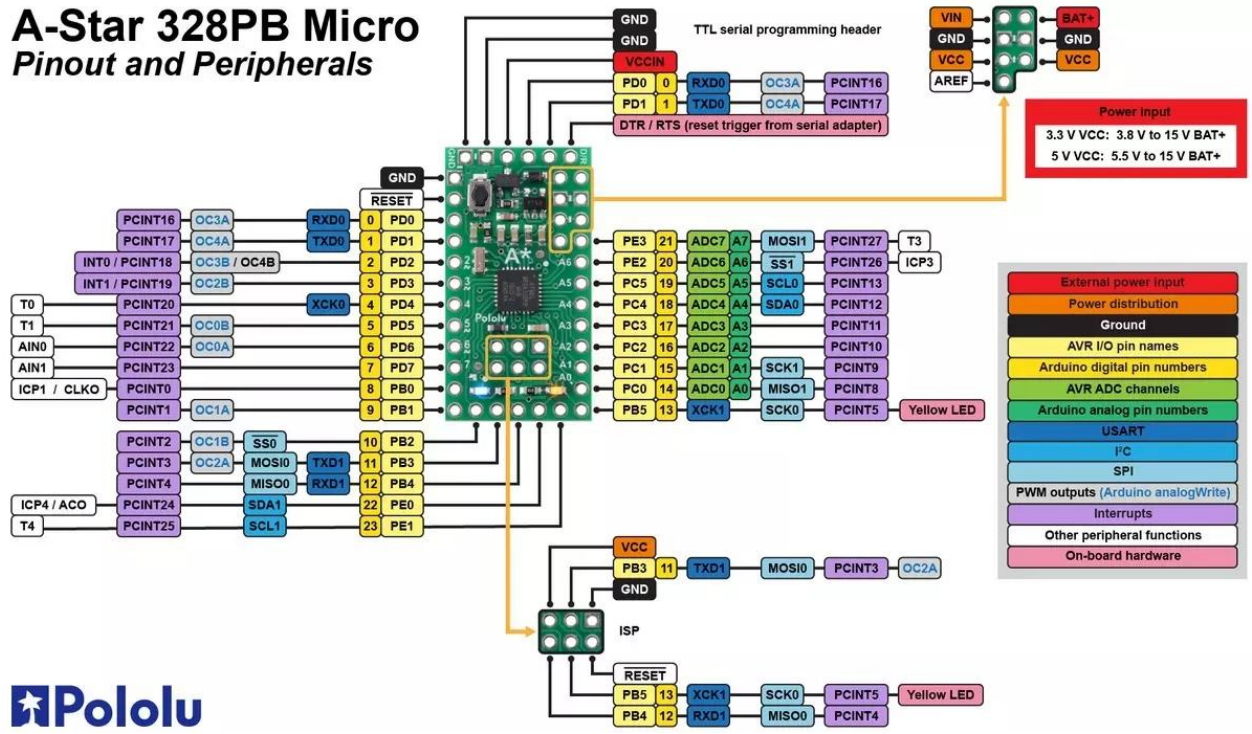




ATmega328	NFR24L01
+3.3V	Vcc
GND	GND
D9	CSN
D10	CE
D11	MOSI
D12	MISO
D13	SCK

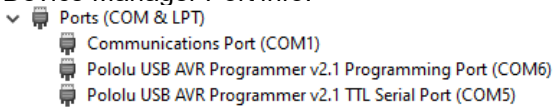
2.5 A* 328PB Micro



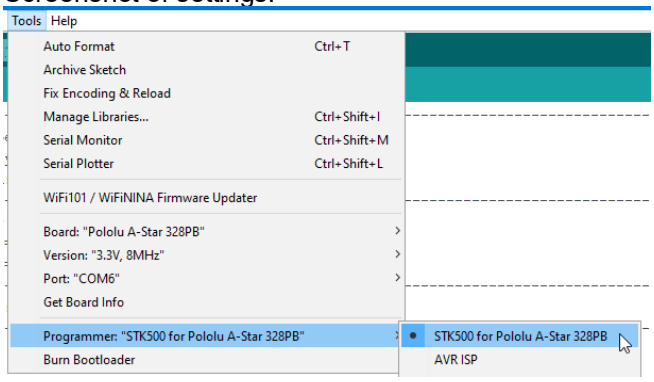


Programming for IDE v2.3.x (for older versions it might be different):
See the additional note below the schematic diagram related to settings.
- The board should be set to "Pololu A-Star 328PB".
  Open “File -> Preferences” and add the next line in “additional board manager URL’s:
  https://files.pololu.com/arduino/package_pololu_index.json
  Note: If items exist, then add the line above on a new line. Use the icon after the input line to do multi-line input.
  Install the boards by using the board manager pane.
- The version should be set correctly depending on the used board.
- The port should be set to the port 'n' connected to "....TTL Serial Port (COM n)".
- The programmer should be set to "Atmel STK500 development board".
- The programming board does not supply the CPU board. Provide a supply to the CPU board.

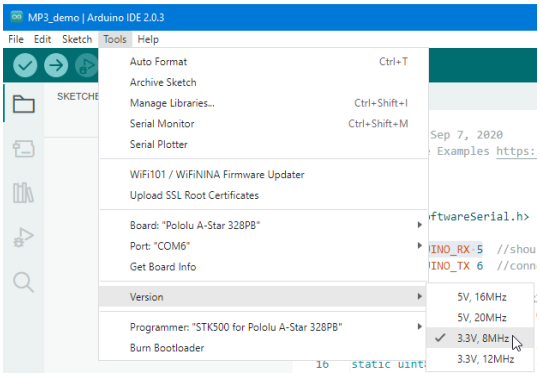
Device Manager Port info:



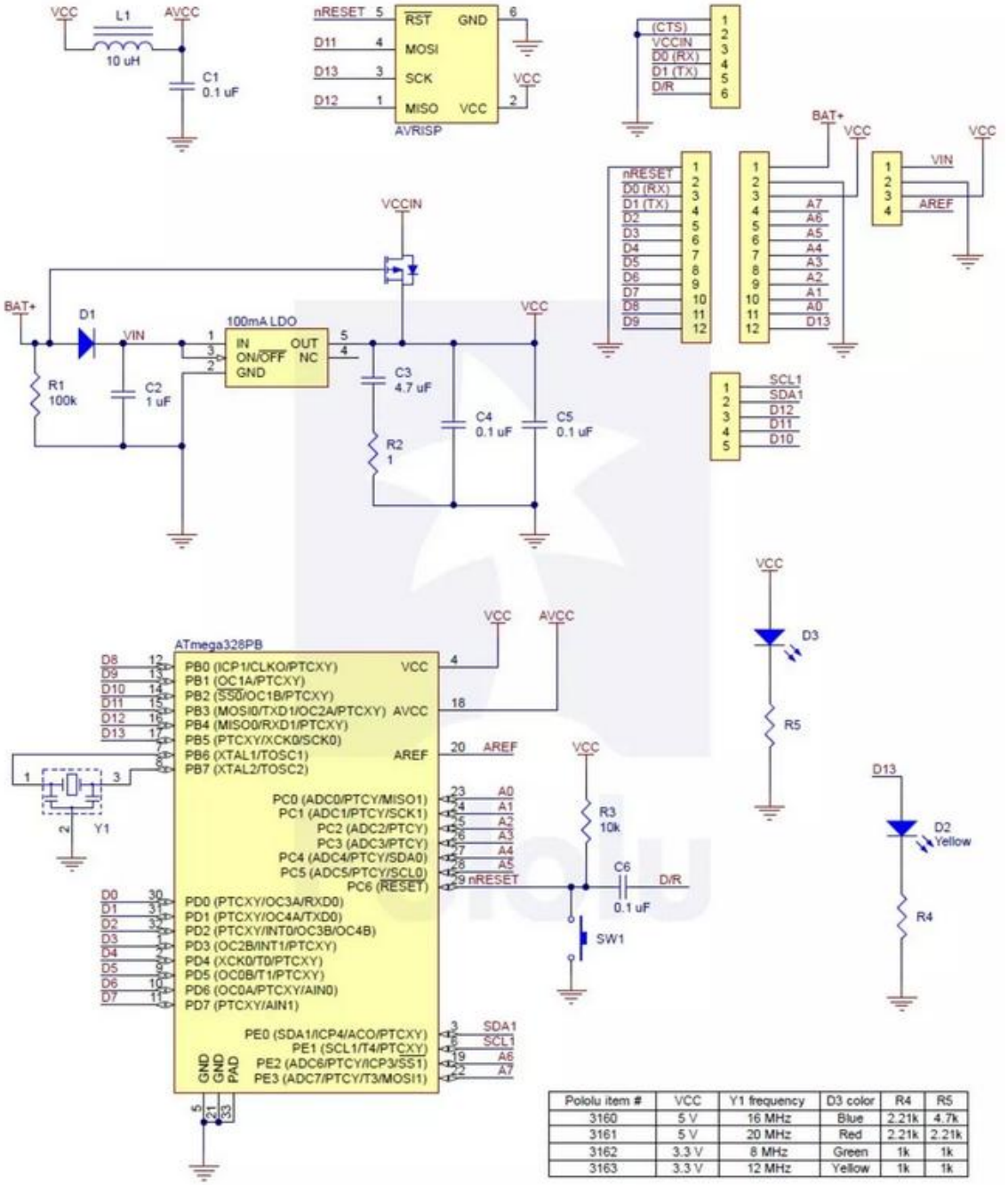
Screenshot of settings:



Board version:



Schematic: (Including table D3 color led indicates the board type related to supply and frequency)



Pololu item #	VCC	Y1 frequency	D3 color	R4	R5
3160	5 V	16 MHz	Blue	2.21k	4.7k
3161	5 V	20 MHz	Red	2.21k	2.21k
3162	3.3 V	8 MHz	Green	1k	1k
3163	3.3 V	12 MHz	Yellow	1k	1k

For the 3160/1 (5V0) versions the VBat should be between 5.5V and 15V.
For the 3162/3 (3V3) versions the VBat should be between 3.8V and 15V.

Info about sleep mode:
https://developerhelp.microchip.com/xwiki/bin/view/products/mcu-mpu/8-bit-avr/structure/sleep/

The board is programmed via a Pololu programmer. The settings for this programmer have changed over the years, unfortunately, because this takes irregular users a lot of time why suddenly sketches won't program. The setup of this Arduino IDE and programming is not very mature to say the least.

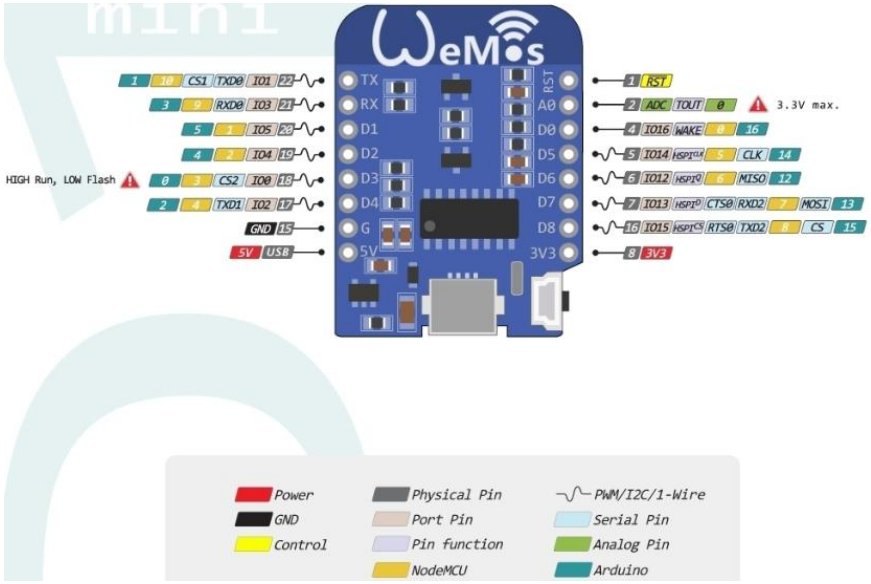
Status 2025 Januari:

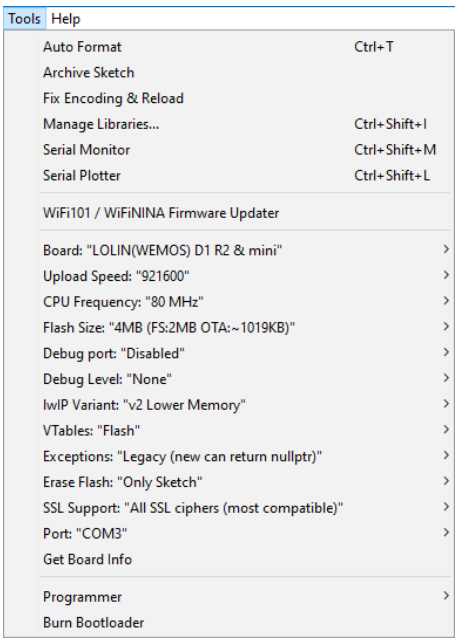
1) PROGRAMMING BOARD:
   Select the programmer in the IDE, this must be: "Atmel STK500 development board"
   Do NOT select the "STK500 for Pololu A-Star 328PB" which was correct in the past, this setting does NOT work !!

2) PORT:
   Select the COM port which corresponds to "Pololu USB AVR Programmer v2.1 TTL Serial Port (COM n)
   Do NOT select the "Programming Port" which was correct in the past, this setting does NOT work !!

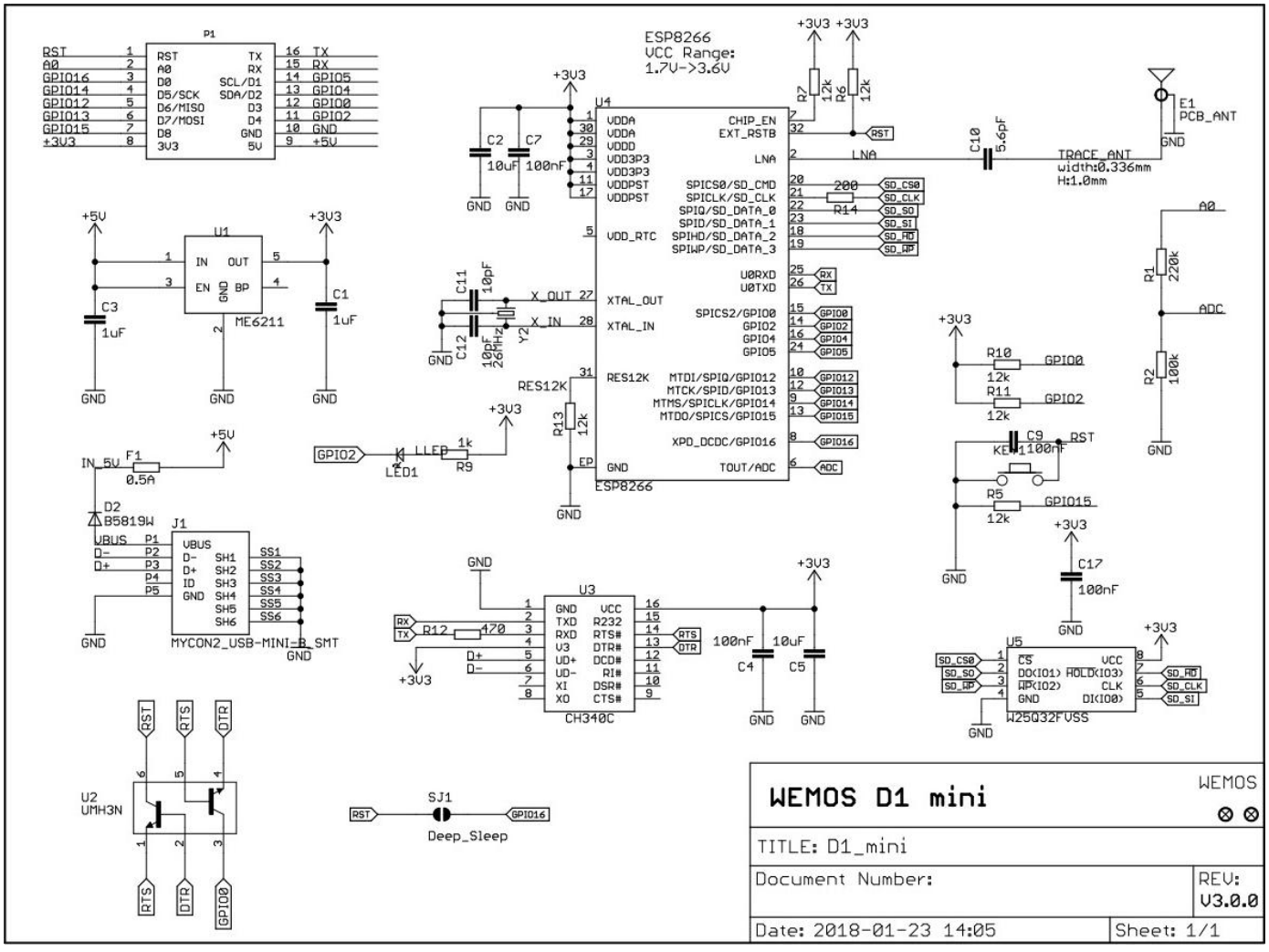
2.6 Wemos D1 Mini

Download the board definition with the Board Manager inside the IDE. See screenshot for programming settings.
Use board: **LOLIN(WEMOS) D1 R2 & mini**

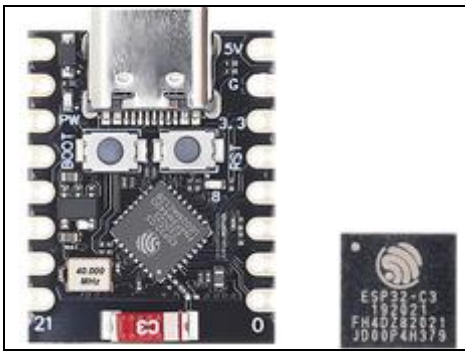




Schematic:



2.7 TENSTAR ROBOT ESP32-C3 Super Mini

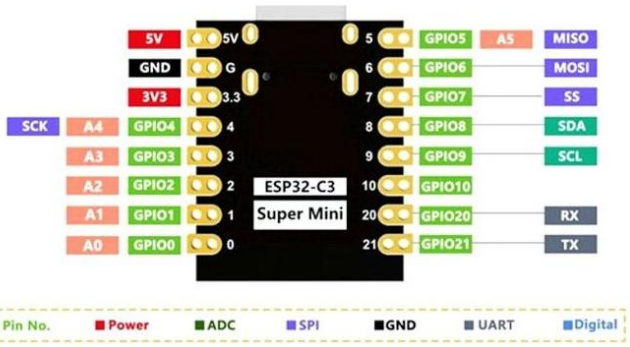


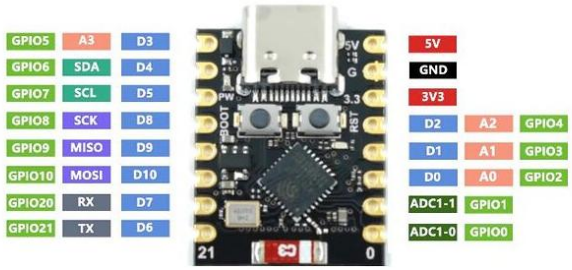
Info:
https://github.com/sidharthmohannair/Tutorial-ESP32-C3-Super-Mini

Programming:
1) Press and Hold “BOOT”
2) Press “RST” shortly (thus press and release)
3) Release “BOOT”

In general: Much higher performance board than (A*/Nano/Mega/…)

- CPU: ESP32-C3, 32-bit RISC-V single core processor @ 160 MHz
- RAM: 400KB, ROM: 374KB
- ADC: 2 * 12-bit SARADC, 6 channels
- WiFi: 802.11b/g/n protocol, 2.4GHz, supports Station mode, SoftAP mode, SoftAP+Station mode, hybrid mode
- Bluetooth: BT 5.0
- Interfaces: 1xI2C, 1xSPI, 2xUART, 11xGPIO (PWM) Levels are 3.3V, 4xADC
- Blue LED, GPIO8 pin





Om de ESP32-C3 Super Mini te programmeren, houdt u eerst de BOOT knop in, dan terwijl u deze inhoud drukt u kort op de RST knop. Daarna kunt u de BOOT knop weer los laten. Als u dit gedaan heeft komt de ESP32-C3 Super Mini in upload modus te staan. U kunt dan via de Arduino IDE uw sketch uploaden.

U kunt uw sketch via de Arduino IDE uploaden door, eerst het board “ESP32C3 Dev Module” te selecteren (Let op: als u deze niet kan vinden, moet u eerst de esp32 boards uit de board manager downloaden), daarna de juiste COM poort te selecteren en dan als laatste uw sketch te uploaden via het pijltje bovenin.

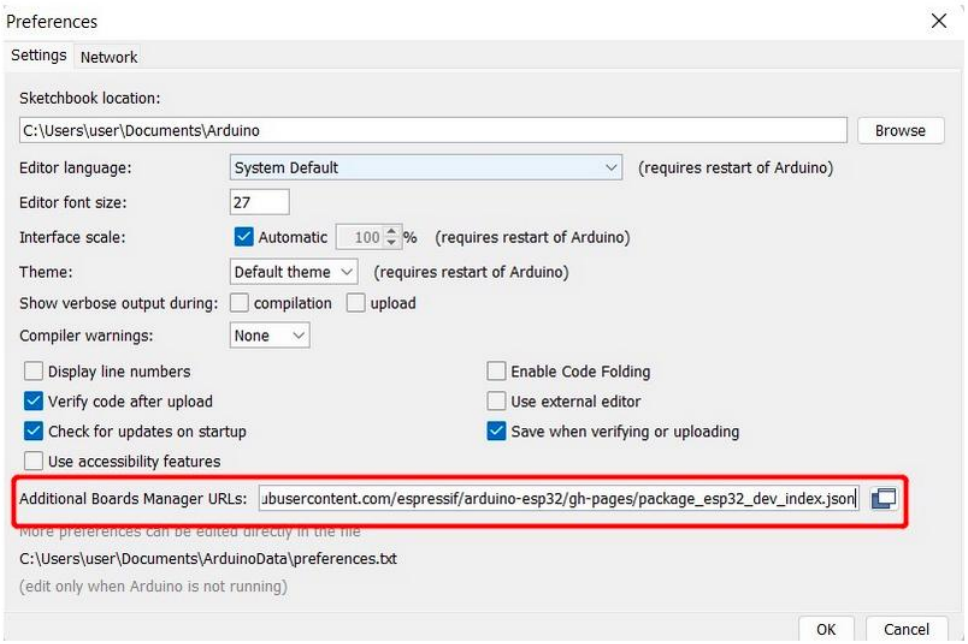
Als uw sketch geüpload is moet u nog een keer op de RST knop drukken. Dit zorgt ervoor dat de ESP32-C3 Super Mini weer uit de upload modus gaat.

Board
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

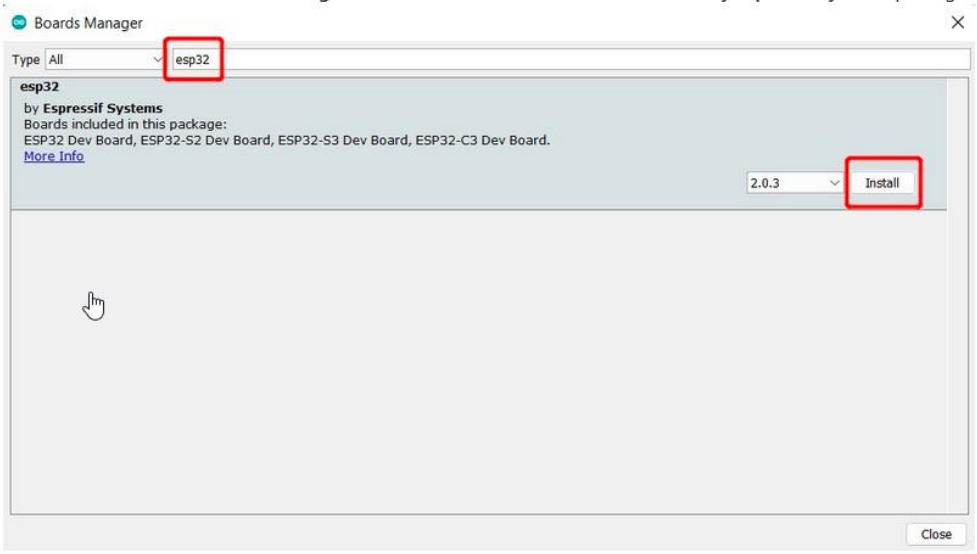
3. Add ESP32 Board Support

1. Open the Arduino IDE and go to **File** > **Preferences**.
2. In the "Additional Board Manager URLs" field, add:

```
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
```



4. Go to **Tools** > **Board** > **Boards Manager**, search for "ESP32," and install the **ESP32 by Espressif Systems** package.



Board info:
The “Tenstar Robot ESP32-C3 Super Mini” is a very small board based on the “Espressif ESP32-C3-MINI-1 module” (same chip: *ESP32-C3*). This is a clone of the “ESP32-C3 MINI” board.
It’s functionally equivalent to:
- LOLIN C3 Mini, or
- ESP32-C3-DevKitM-1, or
- ESP32C3 SuperMini (community nickname)
So you can use **any PlatformIO board definition** that matches that SoC and boot configuration.

PlatformIO
The file platformio.ini is per project and should contain definitions about the board, com interface and wiring.
Here is an example file which works for an ESP32-C3 connected to a TFT display:

```
[env:esp32-c3-devkitm-1]
platform = espressif32
board = esp32-c3-devkitm-1
framework = arduino
monitor_speed = 115200
upload_speed = 921600
```

```
lib_deps =
  moononournation/GFX Library for Arduino@1.4.7

build_flags =
  -D USER_SETUP_LOADED
  -D ILI9488_DRIVER
  -D TFT_SCLK=4
  -D TFT_MISO=5
  -D TFT_MOSI=6
  -D TFT_CS=7
  -D TFT_DC=2
  -D TFT_RST=21
  -D SPI_FREQUENCY=27000000

upload_port = COM6
monitor_port = COM6
```

How to Create a New Project:
1) Open Visual Studio Code
2) In the tab “PIO Home” press the button “New Project”
3) Define the project name
4) Select the board: ESP32-C3-DevKitM-1
5) Select the framework: Arduino

Kind of strange that after opening the project the files from another project are still open in tabs of the GUI, as if they belong to the opened project. This is quite unconvenient, misleading, and absolutely very error prone to start editing a file which is not part of this just opened project. PS: There are many more strange behaviors in the usage of VS Code + PlatformIO, such as the ‘non standard’ way of handling bookmarks. But still a lot better than using vim or command line tools.

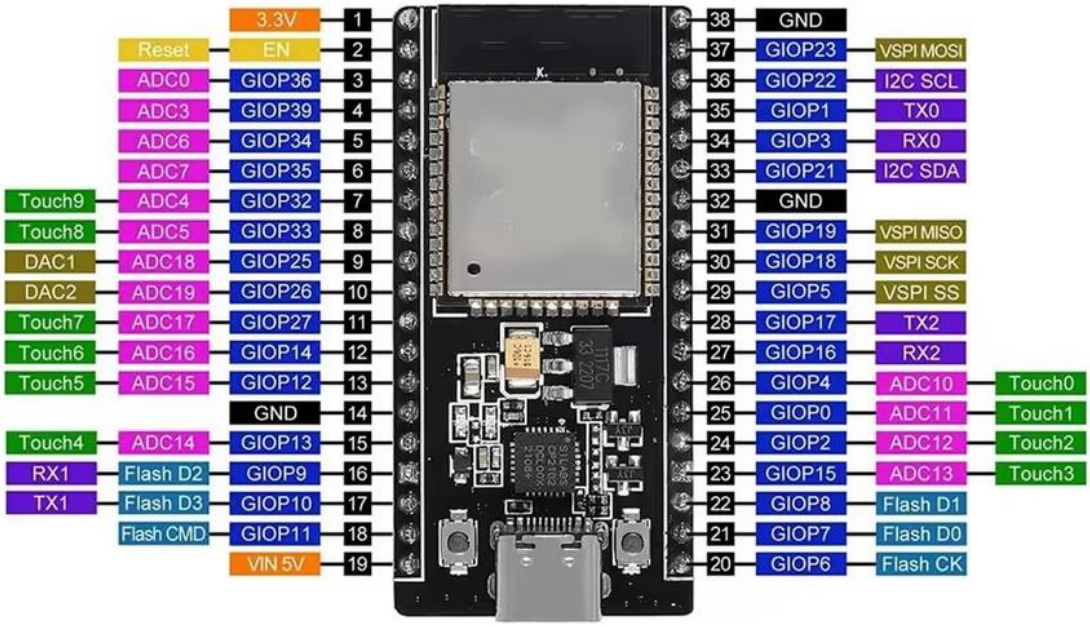
2.8 ESP32 Development Board Type-C CP21023/CH340C Wi Fi + Bluetooth

ESP32-WROOM-32 CP2102 Development Board 32Mbit SPI Flash ESP32 Development Board WiFi Bluetooth-compatible 2412-2484MHz

Development board model:	ESP32-DevKitC-32
Module model:	ESP32-WROOM-32
Main control chip:	ESP32-DOWDQ6-V3 dual-core 32bit MCU integrated WiFi, Bluetooth, Integrated 520-kBSRAM, 448-kBROM16-kBSRAMinRTC
External storage:	4MB
USB driver chip:	CH340C, with good system compatibility, higher download speed, and more stability. Support VIN external wide voltage input 5-12V power supply (battery version maximum 5.5V input) Support USB power supply, external 3.3V power supply, and VIN power supply three kinds of power supply Mode
Board size:	52 x 28 [mm]
Board weight:	10 [g]

Support Arduino IDE mixly, mind+, Python and other programming software

Pinout:



Board in PlatformIO: Espressif ESP32 Dev Module

Example content file Platformio.ini:

```
[env:esp32dev]
platform = espressif32
board = esp32dev
framework = arduino

monitor_speed = 115200
upload_speed  = 921600

monitor_rts = 0
monitor_dtr = 0

monitor_filters = wait_for_upload_port

lib_deps =
        moononournation/GFX Library for Arduino@1.4.7
        adafruit/Adafruit GFX Library@^1.11.9
        paulstoffregen/XPT2046_Touchscreen

upload_port = COM7
monitor_port = COM7
```

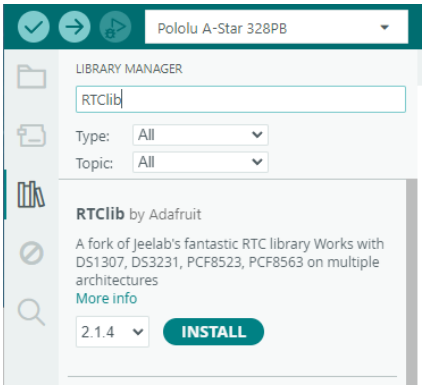
3 Library installations

The recommended method to install libraries is by using the library manager.

Note: The library manager in the Arduino IDE shows official indexed libraries. That means these libraries must contain certain files. On github there are more libraries available. Thus if a library is not part of the library manager, then check github, and see method 2 below.

Method 1: Library manager

1 Open the Arduino IDE
2 Go to Sketch -> Include Library -> Manage libraries
3 Type the library name in the search box, for example: RTClib

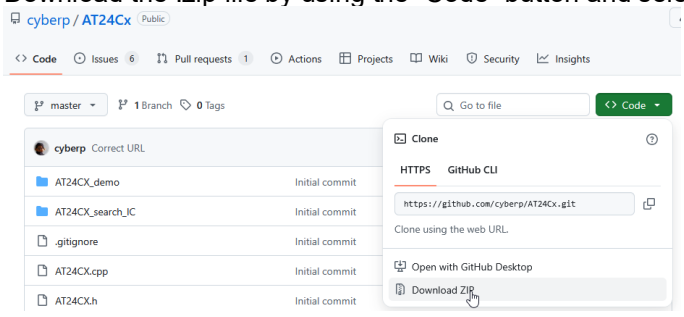


4 Find the library in the list and press INSTALL

Method 2: Install the library zip file

1 Find the library in github.
Example: Search for this in a browser to find the AT24Cx lib: github at24cx
Result: https://github.com/cyberp/AT24Cx

2 Download the .zip file by using the “Code” button and select “Download ZIP”



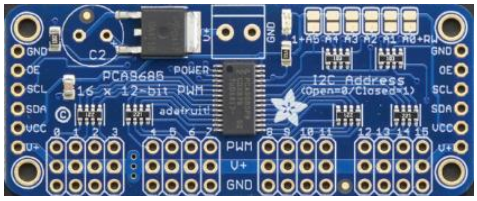
3) In the Arduino IDE open menu: Sketch -> Include library -> Add .ZIP library...
Select the downloaded ZIP file and it will be installed.

Method 3: Manual add (use this as the last resort)

Another method is to download a .zip packaged library from github and uncompress that file.
The complete folder should be added to the library folder which is often “Documents\Arduino\libraries\”.
Then restart the IDE, the Arduino IDE checks libraries during startup.

4 Extension Boards

4.1 Adafruit PCA9685 16 channel Servo Driver



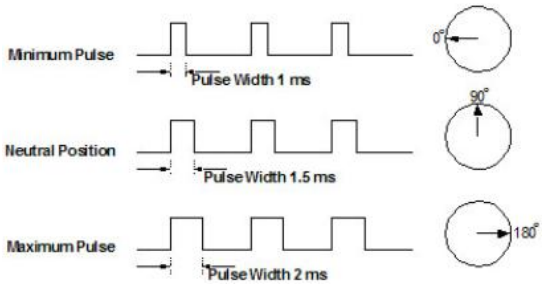
The PWM output pins can drive max 25 mA.

Link to Arduino library: https://learn.adafruit.com/16-channel-pwm-servo-driver/using-the-adafruit-library
Github location: https://github.com/adafruit/Adafruit-PWM-Servo-Driver-Library
Library Documentation: Link

GND	Ground
OE	Output Enable: 0=all pins are enabled, 1=all pins are disabled. Note: Thus the name is wrong, OEn
SCL	I2C Serial clock, has week pull up to Vcc
SDA	I2C Serial data, has week pull up to Vcc
VCC	Logic power, should be same level as control board, between 3V and 5V
V+	Supply for servo's

Board address:
The base address is 0x40. In the upper right corner of the board solder jumpers for [A5:A0] are located. The value of these jumpers is added to the base address. A closed jumper is a logical '1'.

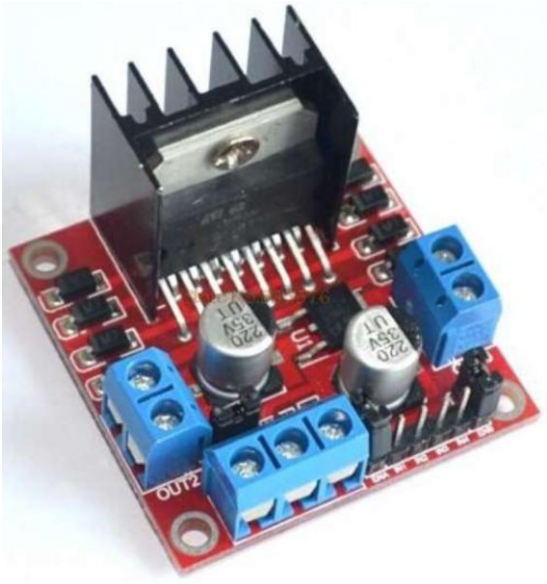
servo motor timing:
Frequency: 50Hz, T=20ms
Pulse: 1 to 2ms

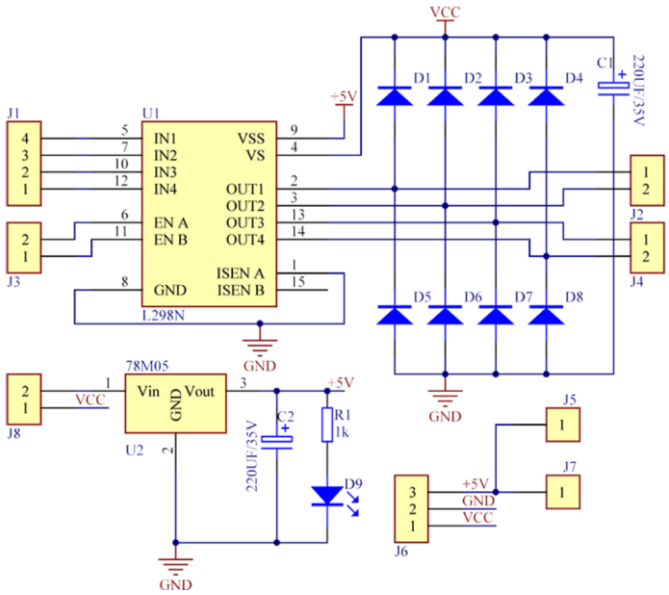


When the frequency is set to 20ms and we want to generate a pulse of t [ms] width, then the PWM value is:
pwmValue = (4096/20) * t = 204.8 * t

```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>

// called this way, it uses the default address 0x40
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();

void setup() {
  Serial.begin(9600);
  pwm.begin();
  pwm.setOscillatorFrequency(27000000);  // The int.osc. is closer to 27MHz
  pwm.setPWMFreq(50);

  // 'fast 400khz I2C' mode
  Wire.setClock(400000);
}

void loop() {
  while (1) {
    for (float t=1.0; t<2.0 ; t+=0.01) {
      uint16_t pwmValue = (int)(204.8*t);
      pwm.setPWM(15, 0, pwmValue);
      delay(50);
    }
  }
}
```

4.2 L298N Dual Motor Controller





Control pins:

Pin	Function
1	EN1 Enable of motor 1 (OUT2 connector) PWM signal for motor 1 speed
2	IN1 (control of motor on connector OUT2)
3	IN2 (control of motor on connector OUT2)
4	IN3 (control of motor on connector OUT3)
5	IN4 (control of motor on connector OUT3)
6	EN2 Enable of motor 2 (OUT3 connector) PWM signal for motor 2 speed

IN1 / IN3	IN2 / IN 4	Function
0	0	Brake
0	1	Motor in direction 1 CW (depends on connection of motor)
1	0	Motor in direction 2 CCW (depends on connection of motor)
1	1	Brake

https://arduinogetstarted.com/tutorials/arduino-controls-stepper-motor-using-l298n-driver

4.3 WEMOS D1 mini DSP8266 Matrix LED Shield 8x8



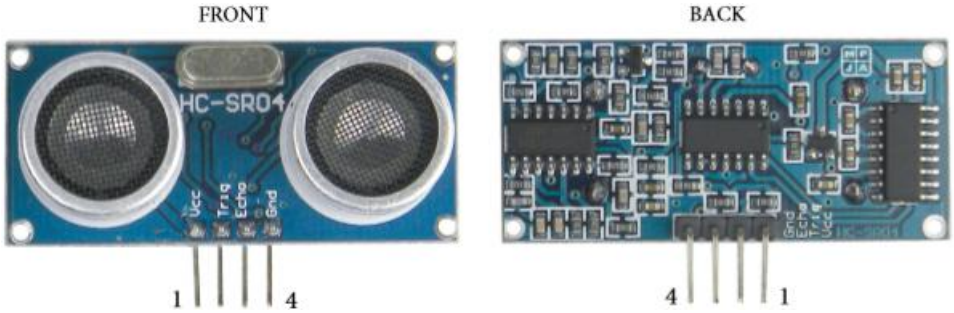
Github location: https://github.com/wemos/WEMOS_Matrix_LED_Shield_Arduino_Library

WEMOS_Matrix_LED.h

```
class MLED {
	public:
		//MLED(uint8_t _intensity=4, byte dataPin=D7, byte clockPin=D5);
		MLED(uint8_t _intensity=4, byte dataPin=11, byte clockPin=10);
		void display();
		void clear();
		void dot(uint8_t x, uint8_t y, bool draw=1);
		void setX(uint8_t x, uint8_t y);
		void barX(uint8_t pos, uint8_t height);
		void barY(uint8_t pos, uint8_t height);

		volatile uint8_t disBuffer[8]={0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
		uint8_t intensity;

	protected:
		void sendCommand(byte led);
		void sendData(byte add, byte data);
		void send(byte data);

		byte dataPin;
		byte clockPin;
};
```

4.4 HC SR-04 Ultrasonic Distance Sensor

Library: Sonar_HC_SR04
Github: https://github.com/HenkBeekhof/Sonar_HC_SR04

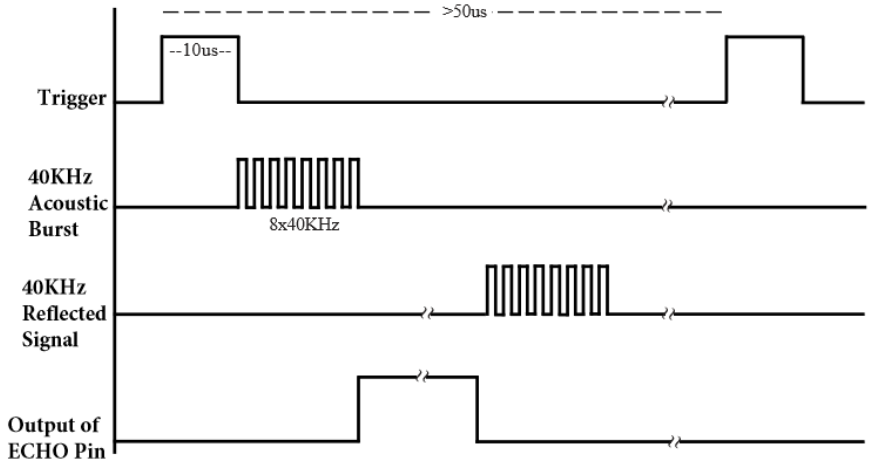


	Pin Symbol	Pin Function Description
1	VCC	5V power supply
2	Trig	Trigger Input pin
3	Echo	Receiver Output pin
4	GND	Power ground

Electrical spec:

Electrical Parameters	HC-SR04 Ultrasonic Module
Operating Voltage	5VDC
Operating Current	15mA
Operating Frequency	40KHz
Max. Range	4m
Nearest Range	2cm
Measuring Angle	15 Degrees
Input Trigger Signal	10us min. TTL pulse
Output Echo Signal	TTL level signal, proportional to distance
Board Dimensions	1-13/16" X 13/16" X 5/8"
Board Connections	4 X 0.1" Pitch Right Angle Header Pins

Timing Diagram:



Operation:
Set 'Trig' and 'Echo' Low to initialize the module.

Place a minimum 10us High level pulse on the 'Trigger' input. The module will automatically send eight 40kHz acoustic bursts. At the same time, gate the microcontroller timer to start timing. Wait to capture the rising edge

output of the 'Echo' pin to stop the timer. Now read the time of the counter, which is the ultrasonic propagation time in the air.

According to the formula: Distance = 'Echo' high level time X ultrasonic velocity (Speed of Sound in air = 340 m/s) / 2, you can calculate the distance to the obstacle.

x = t[s] * 340[m/s] / 2 = t[s] * 170 [m/s] when t in us: x = 170/1000000 * t [m] = 170/10000 * t [cm] = t / 59 [cm]
x = t / 59 [cm], t in [us]

Note:
The Ultrasonic library from the library manager does not work, it gives strange values. This is because the function "micros()" is used, which is unreliable. It turned out that two adjacent reads in the same routine, sometimes gives a lower value on the second read, which is theoretically impossible. This is the source of the wrong results.
The source code is changed and it uses the function "pulseIn()", which does not use the "micros()" function.

Sonar_hc_sr04.h

```
#ifndef Sonar_hc_sr04_h
#define Sonar_hc_sr04_h

// Distance x = t[s] * 340[m/s] / 2 = t[s] * 170 [m/s]
// when t in us: x = 170/1000000 * t [m]  = 170/10000 * t [cm] = t / 59 [cm]
#define CM_FACTOR 59

class Sonar_hc_sr04 {
  public:
    Sonar_hc_sr04(byte trigPin, byte echoPin);
    unsigned int Read(byte factor = CM_FACTOR);

  private:
    byte trig;
    byte echo;

    unsigned long Timing();
};

#endif
```

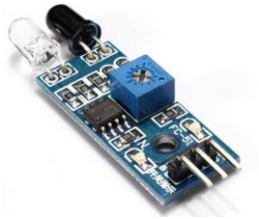
Sonar_hc_sr04.cpp

```
#include "Sonar_hc_sr04.h"

// Constructor: Define the trig and echo pin location and set the direction
Sonar_hc_sr04::Sonar_hc_sr04(byte trigPin, byte echoPin) {
  trig = trigPin;
  echo = echoPin;

  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
}

// Send a 10us pulse on the trig pin and measure the echo pin by using function 'pulseIn'
unsigned long Sonar_hc_sr04::Timing() {
  unsigned long duration;

  // Send a 10us pulse on the Trig pin
  digitalWrite(trig, LOW);
  delayMicroseconds(2);
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);

  // Measure the duration of the high pulse on the Echo pin in microseconds
  duration = pulseIn(echo, HIGH);

  return duration;
}

// Read the distance, result is the distance in centimeters
unsigned int Sonar_hc_sr04::Read(byte factor) {
  return (Timing() / factor);
}
```

Example:

```
#include <Sonar_hc_sr04.h>

Sonar_hc_sr04 Sonar_Front(16, 17);
int distance;

void setup() {
  Serial.begin(115200);
}

void loop() {
  distance = Sonar_Front.Read();

  Serial.print("Distance in [cm]: ");
  Serial.println(distance);
  delay(500);
}
```

4.5 IR sensor module



Obstacle in range: Output is logical '0'
No obstacle in range: Output is logical '1'

Range can be set by the pot-meter on the module.

When obstacle in range, LED turns on.

4.6 Servo



The default available 'Servo' library can be used to control servo motors.
Documentation: https://www.arduino.cc/en/reference/servo

Related to the RAMPS board: Servo's are connected to AUX-X1.
Note: The supply of +5V is not connected. To use this connector, add a jumper between pin VCC and +5V, these are the inner two pins of the 3 pins left of the reset button of the RAMPS board.

Example:

```
const byte SERVO_PIN = 4;

#include <Servo.h>
Servo S1;

void setup() {
  S1.attach(SERVO_PIN);
  S1.write(90);  // mid position
}
void loop() {
}
```

This servo library is a lot of code, there is not much documentation about used resources and how the actual PWM signals are generated.
In the datasheet of the CPU the timers are described. Below code is valid for the A* board running at 8MHz, for other boards, other system clock speed may apply which affects the timers. Below code controls 2 servos based on an 16 bit counter and two servos based on an 8 bit counter. Settings are described in comment. This 9 lines of code 'replaces' the servo library and one knows exactly which resources are used.

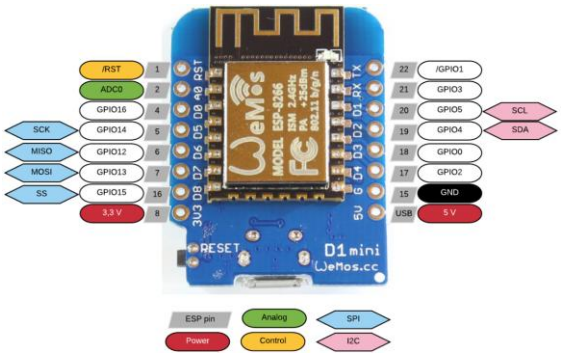
```
// Setting for Timer 1 (16 bits), used for pin PB1 and PB2
```

```
// PWM period time of ~10 ms is wanted
// Fclk_io = 8 MHz, Fclk_timer should be 1 MHz, one tick equals 1 us. => Prescaler = 8, CS[2:0]=010
// The timer waveform generation mode is Fast PWM, TOP=ICR1. => Mode 15, WGM1[3:0]=1110
// The timer counts up from BOTTOM to TOP. TOP is defined by ICR1 (16 bit value) => Tpwm = 10 ms period: ICR1=10000
// The timer mode is to Clear OC1x on compare match and Set on BOTTOM. => COM1A[1:0]=10 COM1B[1:0]=10
TCCR1B=(1<<CS11)|(1<<WGM13)|(1<<WGM12);
TCCR1A=(1<<WGM11)|(2<<COM1A0)|(2<<COM1B0);
// Setting for Timer 0 (8 bits), used for pin PD5 and PD6
// PWM period time of ~10 ms is wanted
// Fclk_io = 8 MHz => T=32us 0xFF*32us=8.192ms Prescaler = 256, CS[2:0]=100
// The timer waveform generation mode is Fast PWM, TOP=0xFF. => Mode 15, WGM0[2:0]=011
// The timer mode is to Clear OC0x on compare match and Set on BOTTOM. => COM0A[1:0]=10 COM0B[1:0]=10
TCCR0B=(1<<CS02);
TCCR0A=(3<<WGM00)|(2<<COM0A0)|(2<<COM0B0);
// Setup timer's period time to 20ms (20ms/Ttimer=10000d)
ICR1=10000;
// Set the servos is the mid position
OCR1A=1500;
OCR1B=1500;
OCR0A=47;
OCR0B=47;
```

4.7 WEMOS D1 mini DSP8266 WiFi

Info: https://escapequotes.net/esp8266-wemos-d1-mini-pins-and-diagram/

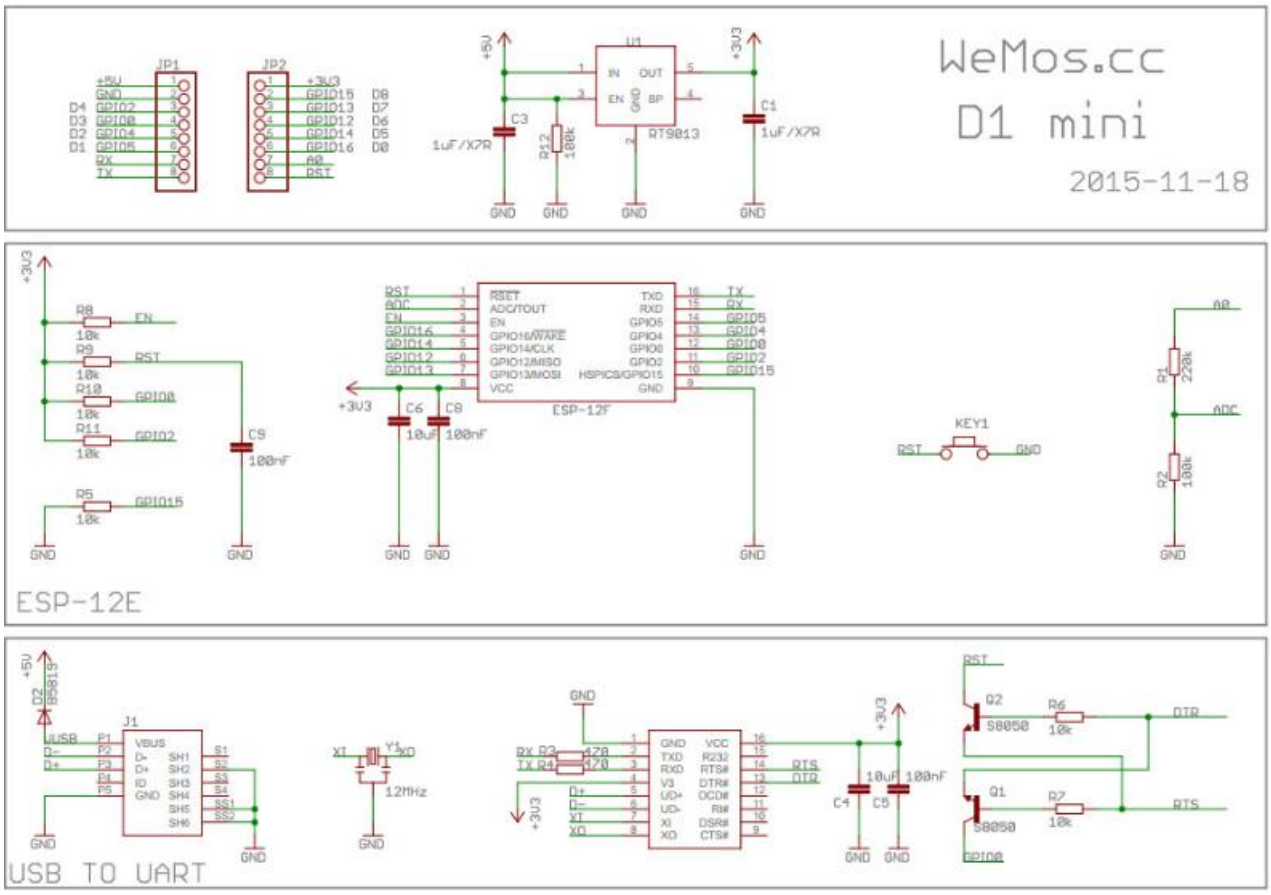
Frontside view:



Pin	Function	ESP-8266 Pin
TX	TXD	TXD
RX	**RXD**	**RXD**
A0	Analog input, max 3.3V input	A0
D0	**IO**	**GPIO16**
D1	IO, SCL	GPIO5
D2	**IO, SDA**	**GPIO4**
D3	IO,10k Pull-up	GPIO0
D4	**IO, 10k pull-up, BUILTIN_LED**	**GPIO2**
D5	IO, SCK	GPIO14
D6	**IO, MISO**	**GPIO12**
D7	IO, MOSI	GPIO13
D8	**IO,10k pull-down, SS**	**GPIO15**
G	Ground	GND
5V	**5V**	-
3V3	3.3V	3.3V
RST	**Reset**	**RST**

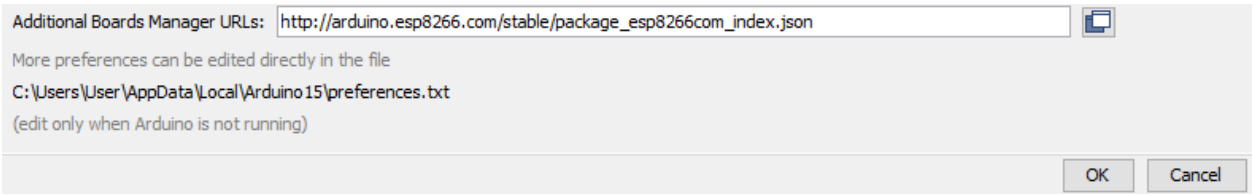
Processor: Espressif ESP8266EX, Tensilica Xtensa LX106 CPU, default 80MHz, can be overclocked to 160MHz.
ESP12-F is a module containing: ESP8266EX CPU, 4MB flash, antenna, shield

Schematics:



Add the board to the Arduino IDE:
File -> Preferences -> “Additional Boards Manager URLs:”
Copy paste the next URL

`http://arduino.esp8266.com/stable/package_esp8266com_index.json`



Tools -> Boards Manager
Select this board and press “Install”.



Set the Board to: Wemos D1 R1

Location of ESP8266WiFi library:
C:\Users\User\AppData\Local\Arduino15\packages\esp8266\hardware\esp8266\2.7.4\libraries\ESP8266WiFi\

Examples\WiFiManualWebServer Web server with page to toggle on board LED on/off

Info:
ESP8266WiFi Library: https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html

4.8 FC16 Matrix LCD (MAX7219)





Module:	FC16
IC:	MAX7219
LCD:	1088AS
8x8LED	
Vcc:	5V
Control:	SPI

Columns and Rows
Asuming the input cable is positioned at the right side, the the following info about columns and rows is valid:
In above diagram the X axis represents Column, starting at 0 at the right side.
Columns: G=0, F=1, DP=7, and the next chained segment G=8, F=9, etc.
Rows: D0 to D7 represent a row. D0 is the LSB of the data.
The triangle of dots in the bottom left corner, the dot in ‘coordinate’ {DP,D4} equals {Col=7,Row=4}

Fonts
Byte 0: The number of column bytes (N) that form this character, equals zero if the character is undefined
Byte 1..N: One byte for each column of the character. Thus it represents the content of a column. The LSB of this byte corresponds to row 7, the MSB corresponds to row 0.

Example: Character ‘&’, represented in 5 bytes:
5, 0x36, 0x49, 0x56, 0x20, 0x50, // 38 - '&'

Column	9	8	7	6	5	4	…	0
Font Byte	1	2	3	4	5			
Row 0, bit 7		■						
Row 1, bit 6	■		■					
Row 2, bit 5	■		■					
Row 3, bit 4		■						

Row 4, bit 3	■		■		■			
Row 5, bit 2	■			■				
Row 6, bit 1		■	■		■			
Row 7, bit 0								
Font value [hex]	36	49	56	20	50			

Library MD_MAX72xx

Font definition
The font is located in file: **MD_MAX72xx_font.cpp**
This is a variable sized font, the first byte of a character specifies the width

```
MD_MAX72XX::fontType_t PROGMEM _sysfont_var[] =
{
  'F', 1, 0, 255, 8,
  0,  // 0 - 'Empty Cell'
  5, 0x3e, 0x5b, 0x4f, 0x5b, 0x3e,  // 1 - 'Sad Smiley'
  5, 0x3e, 0x6b, 0x4f, 0x6b, 0x3e,  // 2 - 'Happy Smiley'
  5, 0x1c, 0x3e, 0x7c, 0x3e, 0x1c,  // 3 - 'Heart'
```

How to use your own font
??

See library documentation on the internet or from the downloaded library:
file:///C:\Users\User\Documents\Arduino\libraries\MD_MAX72XX-master\docs\index.html
Examples to set a point or a byte on a row or column:

```
setPoint(Row, Col, Value);
setRow(RowNr, Value);
setColumn(ColumnNr, Value);
```

How to position a character
setFont(fontType_t* f);

Other definition of fonts: Using a fixed size
For the width of characters a fixed size is used. Setting a pointer to a character is easy, however based on the font definition the order of characters needs to be handled.

This specifies a font named `myfont`, containing 80 characters, where each characters is 5 bytes wide.

```
unsigned const char PROGMEM myfont[80][5] = {
  {0, 0, 0, 0, 0},                  // Space
  {0x3f, 0x48, 0x48, 0x48, 0x3f},   // A
  {0x7f, 0x49, 0x49, 0x49, 0x36},   // B
```

This requires mapping of ascii character value to a position in the font table, example:

```
if (c >= 'A' && c <= 'Z' ) {
    c &= 0x1F;                  // A-Z maps to 1-26
} else if (c == ' ') {
    c = 0;                      // space is located on position 0
} etc. etc.
```

This is one standard mapping routine, the resulting value for c is used in next fetch action:

```
for (char col = 0; col < 5; col++) {
  fontByte = pgm_read_byte_near(&myfont[c][col]);
```

Parola library
See library documentation on the internet or from the downloaded library:
file:///C:/Users/User/Documents/Arduino/libraries/MD_Parola-master/docs/index.html

Definition and instantiation

```
#include <MD_Parola.h>
#include <MD_MAX72xx.h>
#include <SPI.h>

// Define the matrix LCD type and size
#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
#define MAX_DEVICES 4
```

```
#define CLK_PIN   D5
#define DATA_PIN  D7
#define CS_PIN    D8

// Define the display
MD_Parola P = MD_Parola(HARDWARE_TYPE, CS_PIN, MAX_DEVICES);

char curMessage[64];

void setup() {
  P.begin();
  P.displayClear();
  P.print("Text");
  Delay(5000);
  P.displaySuspend(false);
  strcpy(curMessage,"This is text");
  P.displayScroll(curMessage, PA_LEFT, PA_SCROLL_LEFT, 25);
}

void loop() {
  if (P.displayAnimate()) {
    P.displayReset();
  }
}
```

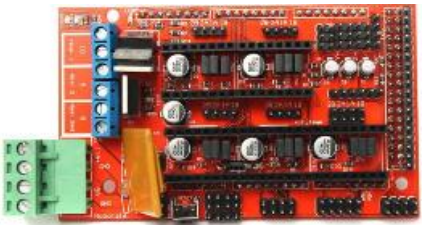
Display a character at a specific location

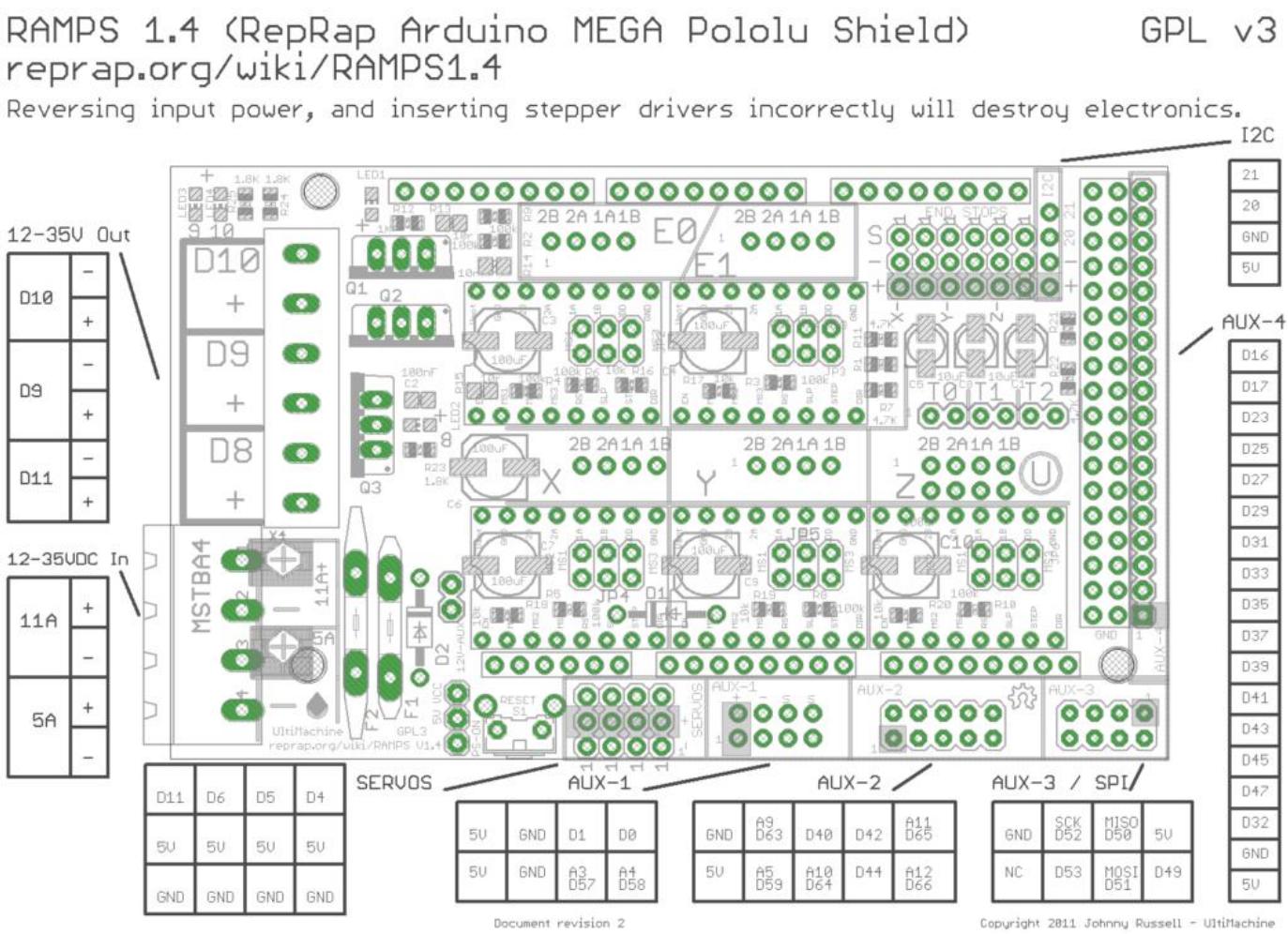
This is not possible with this library. It handles strings, and you can devide the display in several zones, however a zone boundary is a module boundary. It is not possible to create a zone from a certain column to another column. This makes the usage of zones limited and for instance not usable when you want to update specific characters, such as in a counter or time or date.

Writing the whole display over and over again has on the other side a benefit: Disturbance could set unwanted pixels, they stay when that part of the display is not updated.

Thus when individual character positioning is needed, one could consider to use the MD_MAX72xx library without the MD_Parola library.

4.9 RAMPS interface board





Stepper motors: select micro steps and mount the Pololu A4988 driver boards

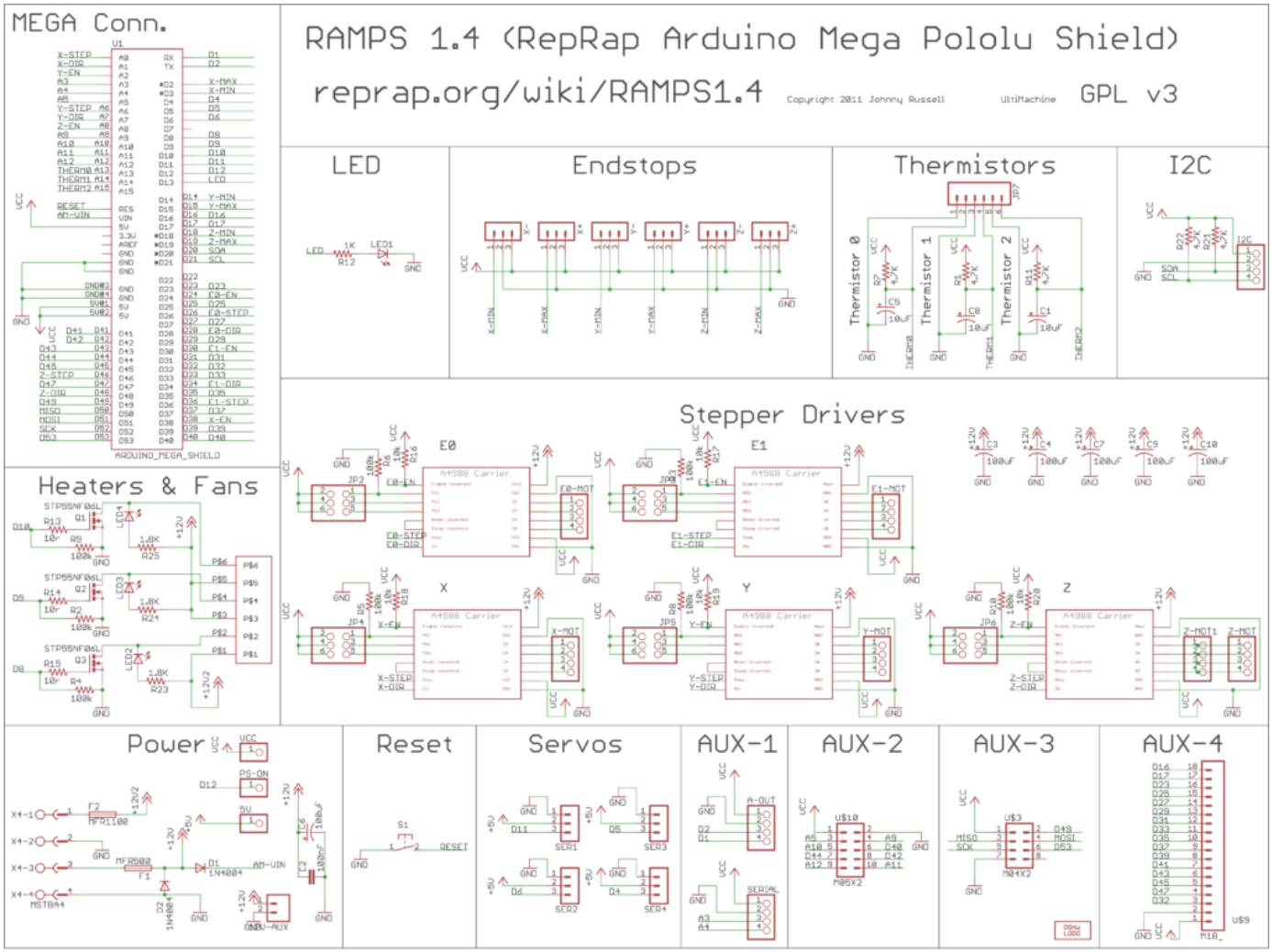
jumper			
1	2	3	Yes/No step size
no	no	no	full step
yes	no	no	half step
no	yes	no	1/4 step
yes	yes	no	1/8 step
no	no	yes	1/16 step
yes	no	yes	1/32 step
no	yes	yes	1/64 step
yes	yes	yes	1/128 step

MS1=Jumper 1 .... MS3=Jumper3 (located beneath the driver boards.
Using a reduced stepper speed reduces the vibrations of the motor.

Sequence: Repeat:{step=1, delay(20), step=0, delay(t)}
Fast motion: t=1200 in full step, equals t=600 in half step, equal t=300 in 1/4 step, equals t=150 in 1/8 step, etc.: meaning this is the same speed of the motor.

Using microstepping reduces motor vibration and has a smoother rotation of the axis.

Schematic:

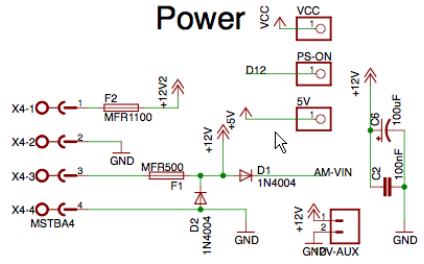


Power supply

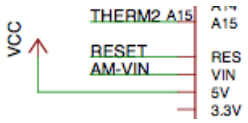
Connect the plus of the power supply to connector 5A, + pin.
Connect the 0V (GND) of the power supply to the 5A, - pin.
It is not required to connect the power to connector 11A. That is only required if connector D8 or D9 is used. Note the error in the layout as shown above, it shows at the upper left corner D10, D9, D11, that should be D10, D9, D8. Mark that D11 is used for the servos.

The Arduino board is powered by the RAMPS power. The connector 5A on the layout is X4 in the schematic, see pin 3 and 4. Pin 3 is connected via a diode D1 to AM-VIN. In the interconnection diagram it is shown that this RAMPS signal AM-VIN is connected to VIN on the Arduino board. The voltage level is 12V minus the diode threshold, so roughly 11.3 Volt. On the Arduino schematics VIN is the input voltage of the 5V regulator. It is not required to supply the Arduino board through the power plug X1.

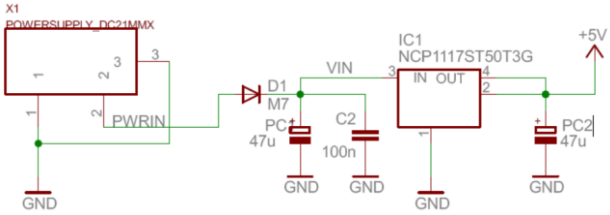
RAMPS V1.4 Schematics Power



Interconnect



Arduino Mega2560 Schematics (Power part)



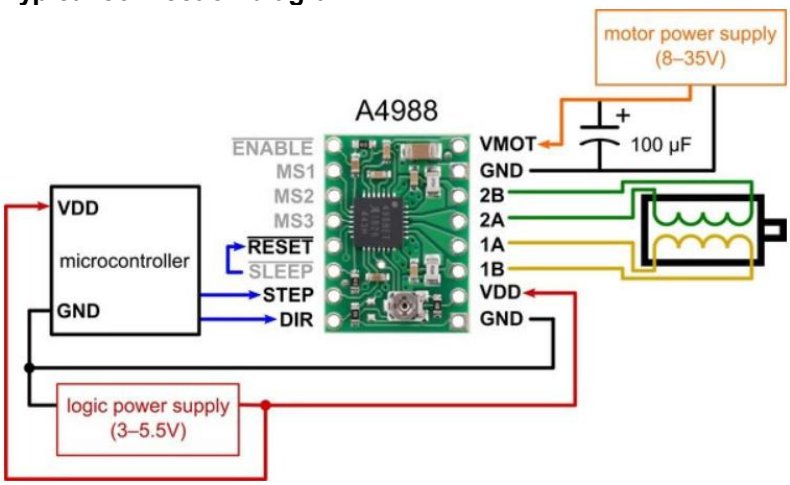
4.10 A4988 Stepper motor driver



See diagram below for pin function. the Elco at VMOT should be close to the board and at least 47uF.

Control inputs
Each pulse to the STEP input corresponds to one micro-step of the stepper motor in the direction selected by the DIR pin. Note that the STEP and DIR pins are not pulled to any particular voltage internally, so you should not leave either of these pins floating in your application. The chip has three different inputs for controlling its many power states: RST, SLP, and EN. For details about these power states, see the datasheet. Please note that the RST pin is floating; if you are not using the pin, you can connect it to the adjacent SLP pin on the PCB to bring it high and enable the board.

Typical connection diagram



Current_limit = Vref / (8 * Rs)
where Rs is the sense resistor on your A4988 board (usually 0.05 Ω or 0.1 Ω).

Assume current limit is 167mA
Vref = 8 * Rs * Current_limit
Vref = 8 * 0.1 * 167mA = 133.6 mV

4.11 A3967 Stepper motor driver (EasyDriver)



This board can control one bipolar (2 coils, 4 wires) stepper motor. It has current control and a voltage regulator which either outputs 3.3V or 5.0V. For this regulator a minimum input voltage of 6V is required. One can also use this board for 5V stepper motors where the internal voltage regulator is not needed and also could not generate 5V. This in contradiction to the board spec on internet which states 6V is the minimum voltage.
One has to look at the chip datasheet:

ELECTRICAL CHARACTERISTICS at T_A = +25°C, V_{BB} = 30 V, V_{CC} = 3.0 V to 5.5V (unless otherwise noted)

Characteristic	Symbol	Test Conditions	Limits			
			Min.	Typ.	Max.	Units
Output Drivers						
Load Supply Voltage Range	V_{BB}	Operating	4.75	–	30	V

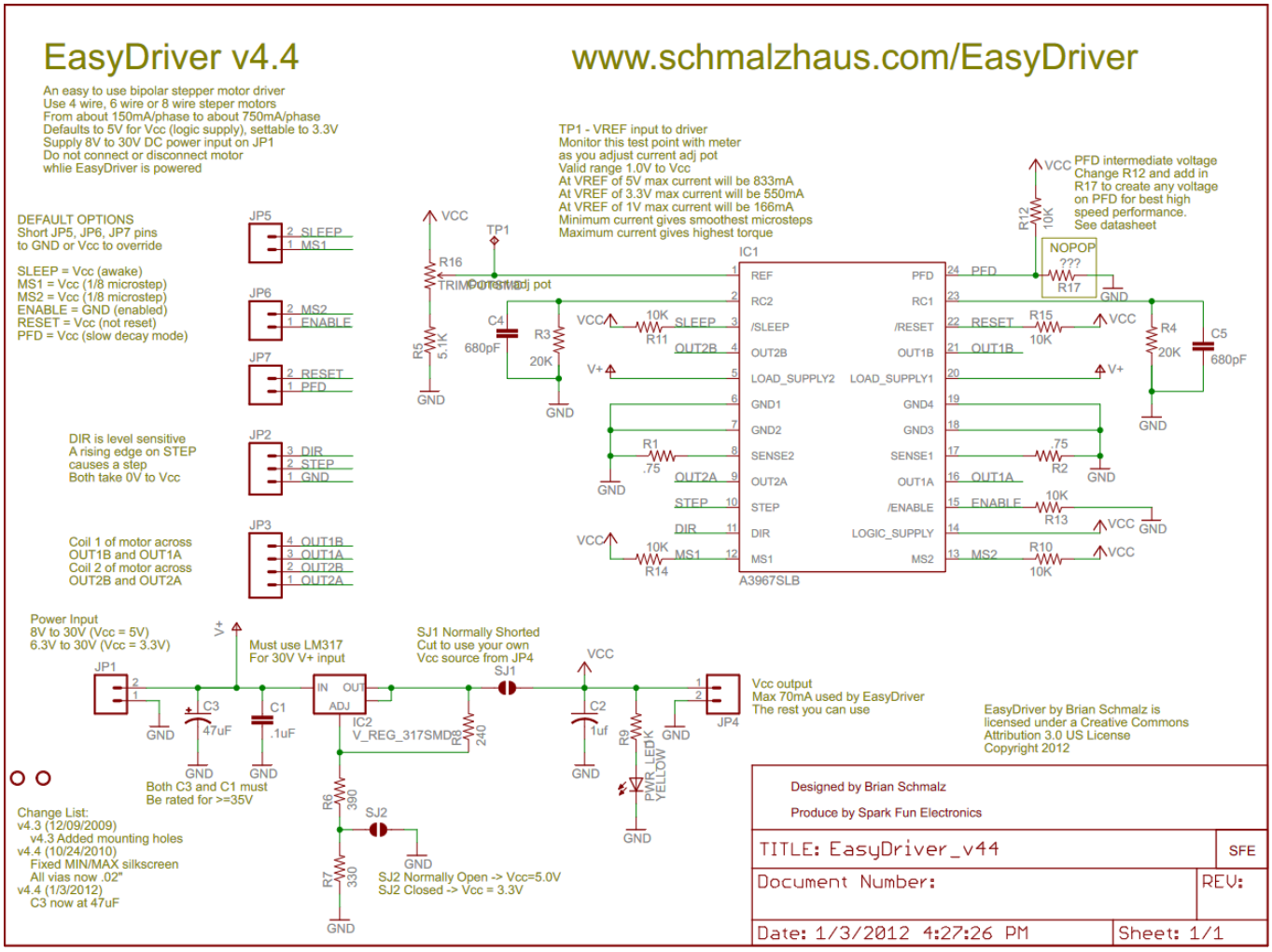
Thus 4.75V is the minimum. Note the remark about Vcc in the table header: 3.0V to 5.5V, what is the spec for the logic levels of the input pins:
Assume the chip is supplied with 5.0V. Vin_high should be minimal 3.5V, this means you cannot drive the logic input pins from a 3.3V source device.

Control Logic						
Logic Supply Voltage Range	V_{CC}	Operating	3.0	5.0	5.5	V
Logic Input Voltage	$V_{IN(1)}$		$0.7V_{CC}$	–	–	V
	$V_{IN(0)}$		–	–	$0.3V_{CC}$	V

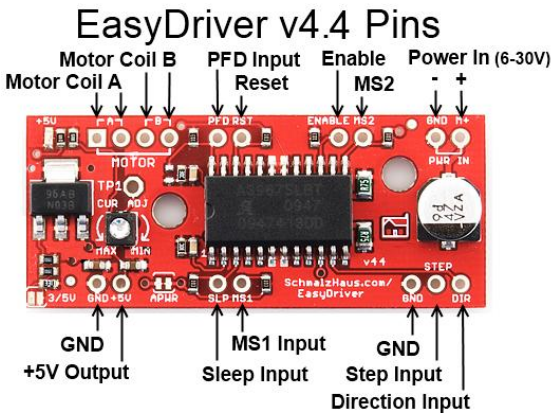
Assume we use a motor on 5.0V, how is the chip supplied?
See schematic on the next page. The Logic supply Vcc of the chip is connected with the output of the voltage regulator. The input of the voltage regulator is JP1 which supplies also the motor via V+. The voltage regulator (LM317) has a recommended headroom (Vi-Vo) of 3V. Meaning the input voltage should be 3V higher than the output voltage.

"Normal case": Assume a motor of 12V or 24V:
JP1=12V or 24V, Usage of the voltage regulator is possible (headroom > 3V).
If the logic which drives the Step and Dir inputs is 5V, then the SJ1 is closed and SJ2 is open, Vcc=5V
If the logic which drives the Step and Dir inputs is 3.3V, then the SJ1 is closed and SJ2 is closed, Vcc=3.3V

"Low voltage motor": Assume a motor of 5V:
The voltage regulator cannot be used, the headroom is too low even for a 3.3V system (5-3.3 < headroom).
SJ2 is irrelevant, SJ1 is open. JP4 is now used as an input instead of an output.
Supply the same voltage as the supply for the logic which drives the Step and Dir inputs to meet Vih requirements.
On a 5V system, JP4 = supplied with 5V.
On a 3.3V system, JP4 = supplied with 3.3V.
The JP1 input should also be supplied with 5V, since this is the V+ input for the chip to drive the motor.



Board connections:

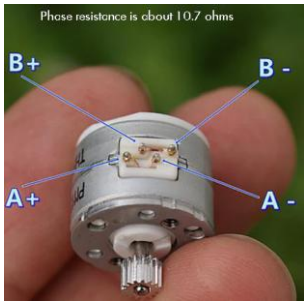


Step resolution settings:

MS1	MS2	Resolution
low	low	Full Step (2 phase)
high	low	Half step
low	high	Quarter step
high	high	Eight step

Source info is found here:
Link: http://www.schmalzhaus.com/EasyDriver/index.html

4.12 Stepper motor PM15S-020





Type:	Micro Stepper motor
Motor diameter:	15mm
Motor height:	ongeveer 11.6mm
Axis diameter:	1.5mm
Output shaft length:	6mm
Coils:	2 coils, bipolar motor
Coil resistance:	~ 10.7 ohm
Step angle:	18 degree
Weight:	~ 9g

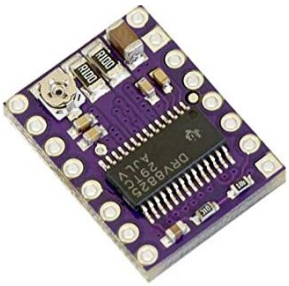
1 revolution: 360/18 = 20 steps per revolution

Max resolution is 8 step microstepping. 1 step = 18/8 = 2.25 degree. 1 revolution = 360/2.25=160 steps (=20*8)

Supply voltage:
PM15S-020-SKH4 5V

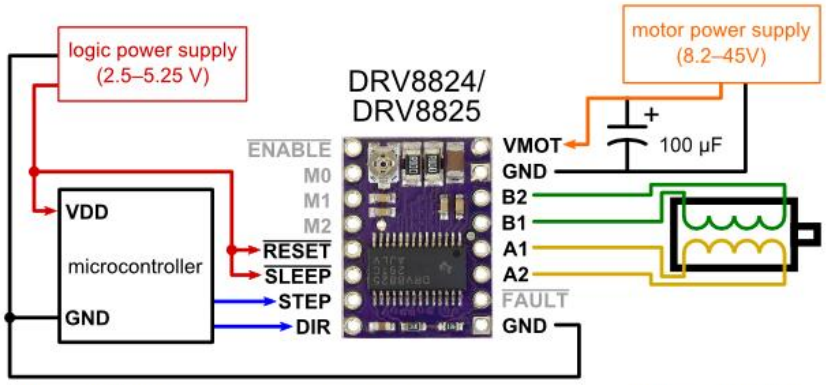
Assuming coil resistance of 30 ohm:
I=U/R => I=5/30=167mA max current.

4.13 DRV8825 Stepper motor driver



Manufacturer: Pololu
Link: https://www.pololu.com/product/2133

Typical connection diagram



Minimal wiring diagram for connecting a microcontroller to a DRV8824/DRV8825 stepper motor driver carrier (full-step mode).

MODE0	MODE1	MODE2	Microstep Resolution
Low	Low	Low	Full step
High	Low	Low	Half step
Low	High	Low	1/4 step
High	High	Low	1/8 step
Low	Low	High	1/16 step
High	Low	High	1/32 step
Low	High	High	1/32 step
High	High	High	1/32 step

Note: The "Typical connection diagram" of the A4988 board can be used as well (board is drop in replacement)

Stepper motors typically have a step size specification (e.g. 1.8° or 200 steps per revolution), which applies to full steps. A microstepping driver such as the DRV8825 allows higher resolutions by allowing intermediate step locations, which are achieved by energizing the coils with intermediate current levels. For instance, driving a motor in quarter-step mode will give the 200-step-per-revolution motor 800 microsteps per revolution by using four different current levels.

The resolution (step size) selector inputs (MODE0, MODE1, and MODE2) enable selection from the six step resolutions according to the table. All three selector inputs have internal 100kΩ pull-down resistors, so leaving these three microstep selection pins disconnected results in full-step mode. For the microstep modes to function correctly, the current limit must be set low enough (see below) so that current limiting gets engaged. Otherwise, the intermediate current levels will not be correctly maintained, and the motor will skip microsteps.

4.14 L78L33A Voltage Regulator 3.3V

Version A = +/- 4% accuracy (Version B = +/- 8% accuracy)

Description:
The L78L series of three-terminal positive regulators employ internal current limiting and thermal shutdown, making them essentially indestructible. If adequate heat-sink is provided, they can deliver up to 100 mA output current. They are intended as fixed voltage regulators in a wide range of applications including local or on-card regulation for elimination of noise and distribution problems associated with single-point regulation. In addition, they can be used with power pass elements to
make high-current voltage regulators. The L78L series used as Zener diode/resistor combination replacement, offers e improvement along with lower quiescent current and lower noise.

Pin connections: (top view, bottom view for TO-92)

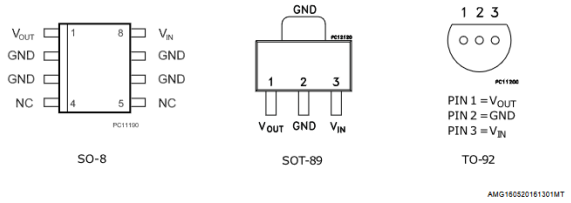


Figure 3. **Test circuits**

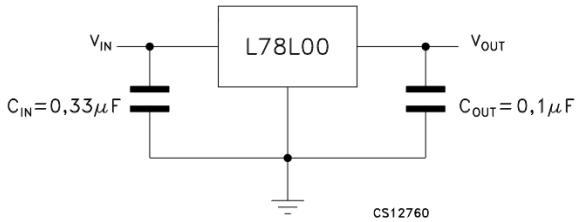


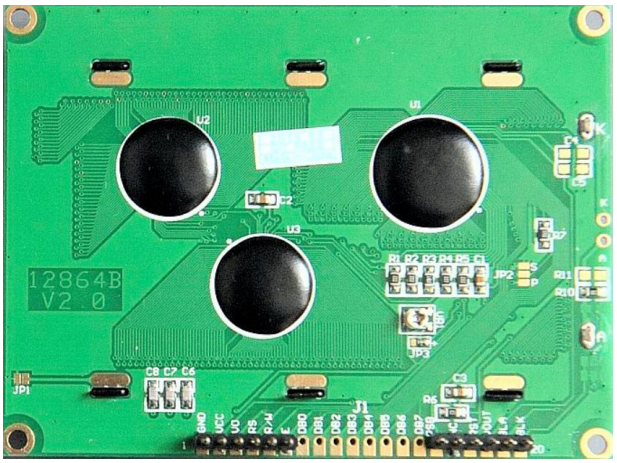
Table 3. **Electrical characteristics of L78L33C - Refer to the test circuits, T_J = 0 to 125 °C, V_I = 8.3 V, I_O = 40 mA, C_I = 0.33 µF, C_O = 0.1 µF unless otherwise specified**

Symbol	Parameter	Test conditions	Min.	Typ.	Max.	Unit
V_O	Output voltage	T_J = 25 °C	3.036	3.3	3.564	V
V_O	Output voltage	I_O = 1 to 40 mA, V_I = 5.3 to 20 V	2.97		3.63	V
		I_O = 1 to 70 mA, V_I = 8.3 V	2.97		3.63	
ΔV_O	Line regulation	V_I = 5.4 to 20 V, T_J = 25 °C			150	mV
		V_I = 6.3 to 20 V, T_J = 25 °C			100	
ΔV_O	Load regulation	I_O = 1 to 100 mA, T_J = 25 °C			60	mV
		I_O = 1 to 40 mA, T_J = 25 °C			30	
I_d	Quiescent current	T_J = 25 °C			6	mA
		T_J = 125 °C			5.5	mA
ΔI_d	Quiescent current change	I_O = 1 to 40 mA			0.2	mA
		V_I = 6.3 to 20 V			1.5	
eN	Output noise voltage	B = 10 Hz to 100 kHz, T_J = 25 °C		40		µV
SVR	Supply voltage rejection	V_I = 6.3 to 16.3 V, f = 120 Hz I_O = 40 mA, T_J = 25 °C	41	49		dB
V_d	Dropout voltage			2		V

4.15 RAMPS Full Grapic Smart Controller

4.15.1 LCD ST7920 controller, 128 x 64 pixels





Library: U8g2
Github: https://github.com/olikraus/u8g2

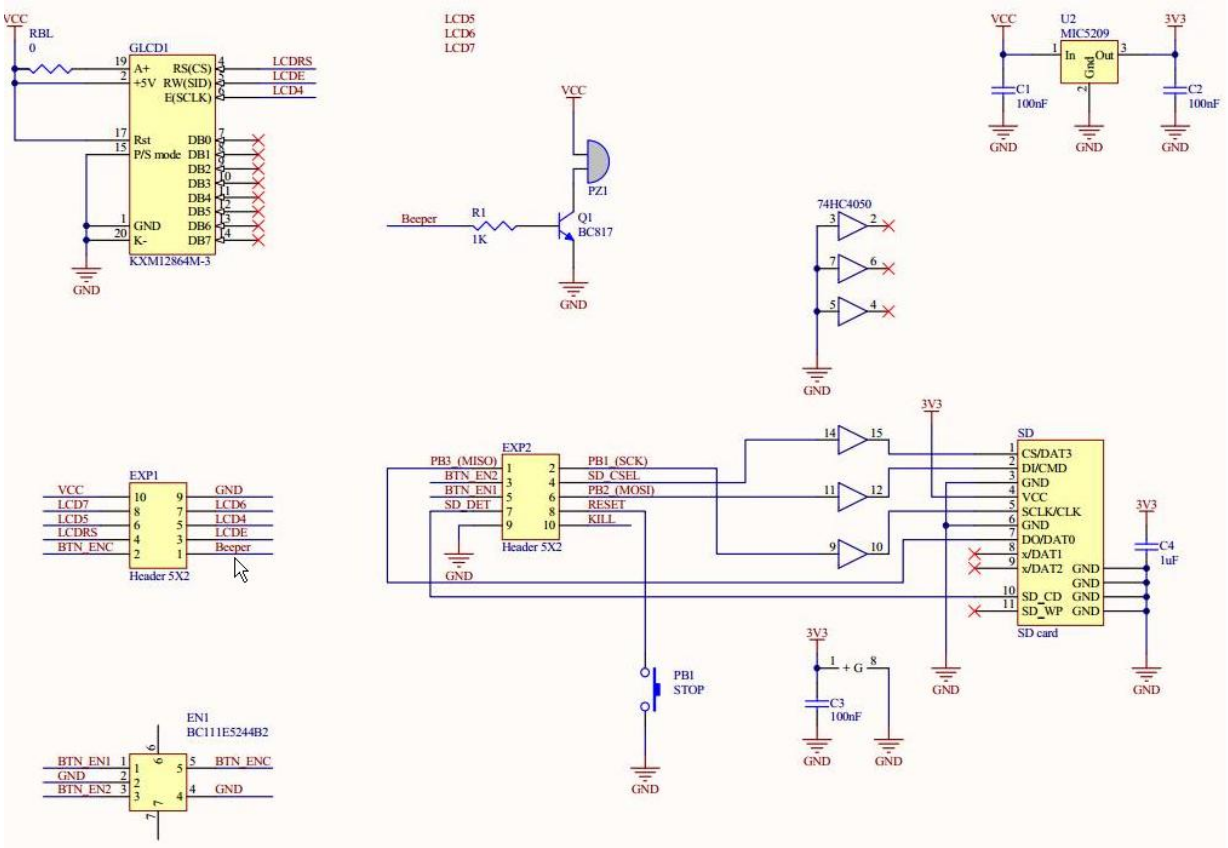
Green board, Backside, pin 1 is defined as leftmost pin.
Green board, Frontside, pin 1 is defined as rightmost pin.

Pin	Signal	
1	GND	0V
2	Vcc	5V
3	Vo	Contrast control
4	RS	SPI: CS
5	R/W	SPI: MOSI
6	E	SPI: Clk
7	DB0	Databus [7:0]
8	DB1	
9	DB2	
10	DB3	
11	DB4	
12	DB5	
13	DB6	
14	DB7	
15	PSB	Paralles/Serial: 0: Serial, 1: Parallel
16	nc	
17	RST	5V
18	Vout	?
19	BLA	Backlight 3.3V
20	BLK	Backlight 0V

Correct setup of U8g2 library:

```
U8G2_ST7920_128X64_1_SW_SPI u8g2(U8G2_R0, /*clock=*/ 23, /*data=*/ 17, /*CS=*/ 16, /*reset=*/ 8);
```

Schematic of Red board: Full Graphic Smart Controller



Connector EXP1

Graphics controller		RAMPS Connector	ATMEGA2560
Pin	Connection		
1	Buzzer	AUX-4: 9	D37
2	Rotary: BTN_ENC	AUX-4: 10	D35
3	LCD: R/W = SPI: MOSI	AUX-4: 17	D17
4	LCD: RS = SPI: CS	AUX-4: 18	D16
5	LCD: E = SPI: Clk	AUX-4: 16	D23
6	nc	AUX-4: 15	D25
7	nc	AUX-4: 14	D27
8	nc	AUX-4: 13	D29
9	GND	AUX-4: 2	GND
10	Vcc	AUX-4: 1	5V

Connector EXP2

Graphics controller		RAMPS Connector	ATMEGA2560
Pin	Connection		
1	SD: DO/ Data0	AUX-3: MISO	D50
2	SD: SCLK	AUX-3: SCK	D52
3	Rotary: BTN_EN2	AUX-4: 12	D31
4	SD: CS/DAT3	AUX-3: D53	D53
5	Rotary: BTN_EN1	AUX-4: 11	D33
6	SD: DI/CMD	AUX-3: MOSI	D51
7	SD: SD_CD	AUX-3: D49	D49
8	Button: Stop	AUX-4: 7	D41
9	GND	nc	
10	nc	nc	

4.15.2 Buzzer

Connector EXP1

Graphics controller		RAMPS Connector	ATMEGA2560
Pin	Connection		
1	Buzzer	AUX-4: 9	D37

The buzzer is connected to D37.

Example:

```
const byte BUZZER_PIN = 37;
unsigned int f;

void setup() {
  pinMode(BUZZER_PIN, OUTPUT);
  f=100;
}
void loop() {
  do {
    tone(BUZZER_PIN, f, 200); // pin, frequency in [Hz], duration in [ms]
    delay(20);
    f+=40;
  } while (f<4000);
  f=100;
}
```

4.15.3 Button

Connector EXP2

Graphics controller		RAMPS Connector	ATMEGA2560
Pin	Connection		
8	Button: Stop	AUX-4: 7	D41

The 'Stop' button is connected to D41
The internal pullup is required for correct operation. The button pressed, connects the button pin to ground, thus by default this pin should be high.

Example:

```
const byte BUTTON_PIN = 41;

void setup() {
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  Serial.begin(115200);
}
void loop() {
  byte b_State = 0;
  while (1) {
    b_State = digitalRead(BUTTON_PIN);
    Serial.println(b_State);
    delay(100);
  }
}
```

4.15.4 Rotary (rotation)

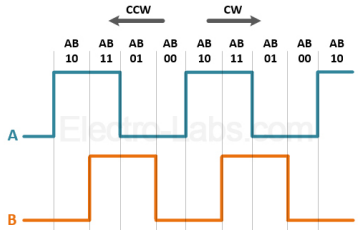
Connector EXP2

Graphics controller		RAMPS Connector	ATMEGA2560
Pin	Connection		
3	Rotary: BTN_EN2	AUX-4: 12	D31
5	Rotary: BTN_EN1	AUX-4: 11	D33

D31 and D33 on the ATMEGA2560:



Physical pins 59 and 57, on ports PC6 and PC4. These pins are not connected to the internal interrupt block, meaning it is not possible to use an interrupt service routine to handle this rotary when using these default connections of the RAMPS interface board.



Rotary.cpp

```
#include "Rotary.h"
// Constructor: Define the two pins, and optinally swap direction
Rotary::Rotary(uint8_t EN1_Pin, uint8_t EN2_Pin, bool swap = false) {
  EN1 = (!swap) ? EN1_Pin : EN2_Pin;
  EN2 = (!swap) ? EN2_Pin : EN1_Pin;

  // Set both pins as input pins, with internal pullup
  pinMode(EN1, INPUT_PULLUP);
  pinMode(EN2, INPUT_PULLUP);

  // Define default values
  A_Previous = digitalRead(EN1);
  B_Previous = digitalRead(EN2);

  rotary_Value = 0;
  rotary_Changed = false;
}
// Reset the rotary value to zero
void Rotary::Reset() {
  rotary_Value = 0;
}
// Check if the rotary value has changed
bool Rotary::Changed() {
        bool current = rotary_Changed;
        rotary_Changed = false;
        return current;
}
// Read the rotary value
int Rotary::Read() {
  A_Current = digitalRead(EN1);
  B_Current = digitalRead(EN2);
  rotary_Changed = false;

  if (A_Current != A_Previous) {
        A_Previous = A_Current;
        rotary_Changed = true;
    if (A_Current != B_Current) {
      rotary_Value++;
    } else {
      rotary_Value--;
    }
  }
  if (B_Current != B_Previous) {
    B_Previous = B_Current;
        rotary_Changed = true;
    if (A_Current == B_Current) {
      rotary_Value++;
    } else {
      rotary_Value--;
    }
  }
  // Debouce: Add 50us delay between reads
  delayMicroseconds(50);
  return rotary_Value;
}
```

Rotary.h

```
#ifndef Rotary_h
#define Rotary_h

class Rotary {
	public:
		Rotary(uint8_t EN1_Pin,  EN2_Pin, bool swap = false);
		void Rotary::Reset();
		int Rotary::Read();
		bool Rotary::Changed();

	private:
		uint8_t EN1, EN2;
		uint8_t A_Previous, A_Current;
		uint8_t B_Previous, B_Current;
		int  rotary_Value;
		bool rotary_Changed;
};
#endif
```

Example:

```
#include <Rotary.h>

const byte ROTARY_EN1_PIN = 33;
const byte ROTARY_EN2_PIN = 31;

Rotary R1(ROTARY_EN1_PIN, ROTARY_EN2_PIN, true);
int R1Value;

void setup() {
  Serial.begin(115200);
}

void loop() {
  R1Value = R1.Read();
  if (R1.Changed()) Serial.println(R1Value);
}
```

4.15.5 Rotary (push button)

Connector EXP1

Graphics controller		RAMPS Connector	ATMEGA2560
Pin	Connection		
2	Rotary: BTN_ENC	AUX-4: 10	D35

The rotary push button is connected to D35
The internal pullup is required for correct operation. The button pressed, connects the button pin to ground, thus by default this pin should be high.

Example:

```
const byte BUTTON_PIN = 35;

void setup() {
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  Serial.begin(115200);
}
void loop() {
  byte b_State = 0;
  while (1) {
    b_State = digitalRead(BUTTON_PIN);
    Serial.println(b_State);
    delay(100);
  }
}
```

4.16 RAMPS Stepper motor

Library: AccelStepper
Location: https://github.com/waspinator/AccelStepper
The stepper motors are controlled by the Pololu controller board. This board has three control pins:
Enable 0=Enabled
Direction Depends on how the motor is connected to the 4 wire connector (2 coils)
Step Pulse to move the motor one step further in the defined direction

4.17 RAMPS pins connections to ATMEGA2560

RAMPS_14_pins.h

```
#ifndef RAMPS_14_PINS_h
#define RAMPS_14_PINS_h

// For RAMPS 1.4 (RepRap Arduino Mega Pololu Shield)
#define X_STEP_PIN         54 // A0
#define X_DIR_PIN          55 // A1
#define X_ENABLE_PIN       38 // D38 0=Enabled
#define X_MIN_PIN           3 // D3  0=End switch closed
#define X_MAX_PIN           2 // D2  0=End switch closed

#define Y_STEP_PIN         60 // A6
#define Y_DIR_PIN          61 // A7
#define Y_ENABLE_PIN       56 // A2  0=Enabled
#define Y_MIN_PIN          14 // D14 0=End switch closed
#define Y_MAX_PIN          15 // D15 0=End switch closed

#define Z_STEP_PIN         46 // D46
#define Z_DIR_PIN          48 // D48
#define Z_ENABLE_PIN       62 // A8  0=Enabled
#define Z_MIN_PIN          18 // D18 0=End switch closed
#define Z_MAX_PIN          19 // D19 0=End switch closed

#define E0_STEP_PIN        26 // D26
#define E0_DIR_PIN         28 // D28
#define E0_ENABLE_PIN      24 // D24 0=Enabled

#define E1_STEP_PIN        36 // D36
#define E1_DIR_PIN         34 // D34
#define E1_ENABLE_PIN      30 // D30 0=Enabled

#define SERVO_1            11 // Servo on pin D11
#define SERVO_2            6  // Servo on pin D6
#define SERVO_3            5  // Servo on pin D5
#define SERVO_4            4  // Servo on pin D4

#define HEATER_FAN1_PIN    8  // D8
#define HEATER_FAN2_PIN    9  // D9
#define HEATER_FAN3_PIN    10 // D10

#define LED_PIN            13 // D13

#define TEMP_0_PIN         13 // A13
#define TEMP_1_PIN         14 // A14
#define TEMP_2_PIN         15 // A15

#define SCL                21 // D21 I2C clock
#define SDA                20 // D20 I2C data

#define ROTARY_EN1_PIN     33 // D33 AUX-4 p11
#define ROTARY_EN2_PIN     31 // D31 AUX-4 p12
#define ROTARY_BUTTON      35 // D35 AUX-4 p10
#define BUZZER             37 // D37 AUX-4 p9
#define BUTTON             41 // D41 AUX-4 p7
#define LCD_MOSI           17 // D17 AUX-4 p17 SPI MOSI=LCD R/W
#define LCD_CS             16 // D16 AUX-4 p18 SPI CS  =LCD RS
#define LCD_CLK            23 // D23 AUX-4 p16 SPI CLK =LCD E
#define SD_CD              49 // D49 AUX-3 D49
#define SD_DO              50 // D50 AUX-3 MISO
#define SD_DI              51 // D51 AUX-3 MOSI
#define SD_SCLK            52 // D52 AUX-3 SCK
#define SD_CS              53 // D53 AUX-3 D53

#define PS_ON_PIN          12 // D12
#endif
```

4.18 HC-05 Bluetooth Serial Interface



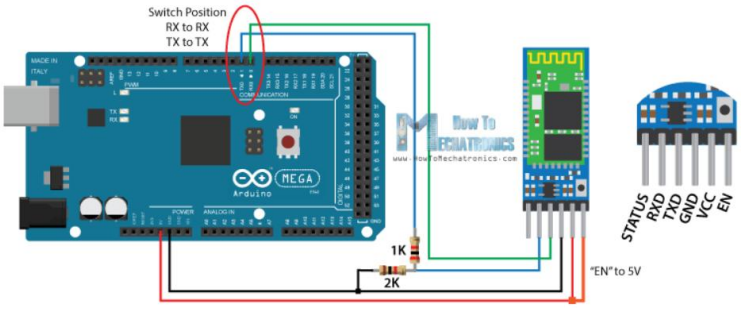
Note: This device is using 3.6V to 5V as power supply, but the interface signal are 3.3V.
See schematic below, a resistor divider is used in front of the HC-05 RX input.

The RX pin should be controllable during downloading code, therefor downloading will not work if below schematic is used. On internet it is mentioned to add a resistor of 4k7 ohm in front of the RX pin of the Arduino board, thus between RX(Arduino) and TX (HC-05). Using this resistor does not work in AT command mode.

https://howtomechatronics.com/tutorials/arduino/how-to-configure-pair-two-hc-05-bluetooth-module-master-slave-commands/

4.18.1 Configuration for Master or Slave

To enter AT mode:
- EN pin should be High.
- RX and TX on the Arduino side should be swapped.
- During power on keep the button above the EN pin pressed.
- Check: Flash every 2 seconds confirms the AT mode is entered.
- Upload empty sketch after disconnecting RX and TX, then connect them in swapped configuration.
- Open monitor program and set baudrate to 38k4, and termination to "Both NL and CR".



AT Command format:

AT+<parameter name>?	Returns current value of the parameter
AT+<parameter name>=<value>	Sets the value of this parameter

Slave configuration, get the address of the slave:

AT	OK	
AT+UART?	+UART:38400,0,0	
AT+ROLE?	+ROLE:0	// Role = 0 is slave mode
AT+ADDR?	+ADDR:nnnn:nn:nnnnnn	

Master configuration:

AT	OK	
AT+UART?	+UART:38400,0,0	
AT+ROLE=1	OK	// Role = 1 is master mode
AT+CMODE=0	OK	// Connect mode = fixed address
AT+BIND= nnnn:nn:nnnnnn	OK	// Set address of the slave device

4.19 TFT color display 320 x 480 ILI9486

https://github.com/ImpulseAdventure/Arduino-TFT-Library-ILI9486
https://www.14core.com/driving-the-qd320db16nt9481ra-3-2-tft-480x320-lcd-ili9481-wd-mega/

Driver: UTFT, HX8357



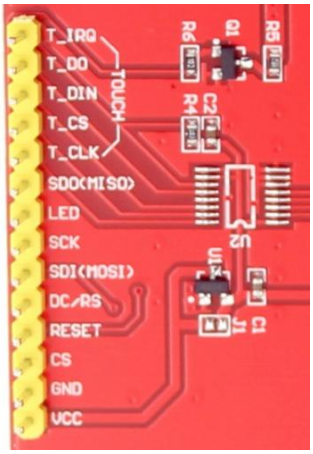
Number	Module Pin	Pin Description	Number	Module Pin	Pin Description
1	5V	Positive power supply	2	5V	Positive power supply
3	DB8	8th bit of data bus	4	DB9	9th bit of data bus
5	DB10	10th bit of data bus	6	DB11	11th bit of data bus
7	DB12	12th bit of data bus	8	DB13	13th bit of data bus
9	DB14	14th bit of data bus	10	DB15	15th bit of data bus
11	DB7	7th bit of data bus	12	DB6	6th bit of data bus
13	DB5	5th bit of data bus	14	DB4	4th bit of data bus
15	DB3	third bit of data bus	16	DB2	2nd bit of data bus
17	DB1	1st bit of data bus	18	DB0	0 bit of data bus
19	RS	LCD register / data selection signal Low level: register, high level: command	20	WR	LCD write control signal
21	CS	LCD screen select control signal, low level enable	22	RST	LCD reset control signal, low reset
23	NC	Undefined, reserved	24	NC	Undefined, reserved
25	NC	Undefined, reserved	26	F_CS	SPI flash chip select control signal
27	NC	Undefined, reserved	28	NC	Undefined, reserved
29	NC	Undefined, reserved	30	NC	Undefined, reserved
31	MISO	SPI bus input signal	32	MOSI	SPI bus output signal
33	CLK	SPI bus clock signal	34	SD_CS	SD card select control signal, low level enable
35	GND	Power ground	36	GND	Power ground

4.20 TFT color display 320 x 480 SPI ILI9488

https://www.lcdwiki.com/3.5inch_SPI_Module_ILI9488_SKU:MSP3520

Note: The VCC is regulated by U1 to 3.3V. The ILI9488 controller chip is NOT 5V tolerant related to input signals.



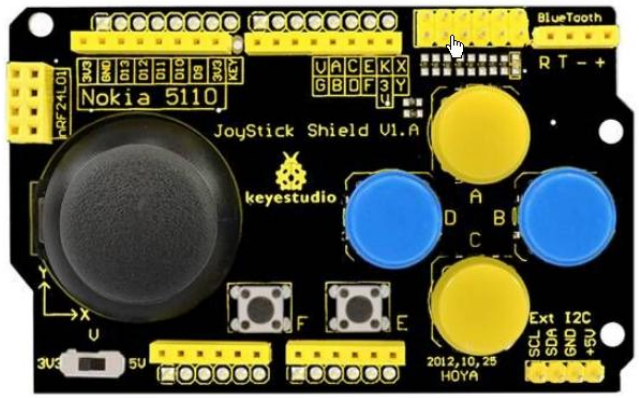


Pin	Name	Description
1	VCC	5V / 3.3V power input
2	GND	Ground
3	CS	LCD chip select signal, low is enabled
4	RESET	LCD reset signal, low is reset
5	DC/RS	LCD register / data select signal, high is register, low is data
6	SDI (MOSI)	LCD SPI write data signal
7	SCK	LCD SPI clock signal
8	LED	Backlight control, high is on, if not controlled then connect to 3.3V (always bright)
9	SDO (MISO)	LCD SPI read data signal
10	T_CLK	Touchscreen SPI clock signal
11	T_CS	Touchscreen chip select signal, low is enabled
12	T_DIN	Touchscreen SPI write data signal
13	T_DO	Touchscreen SPI read data signal
14	T_IRQ	Touchscreen interrupt signal, low when touch is detected

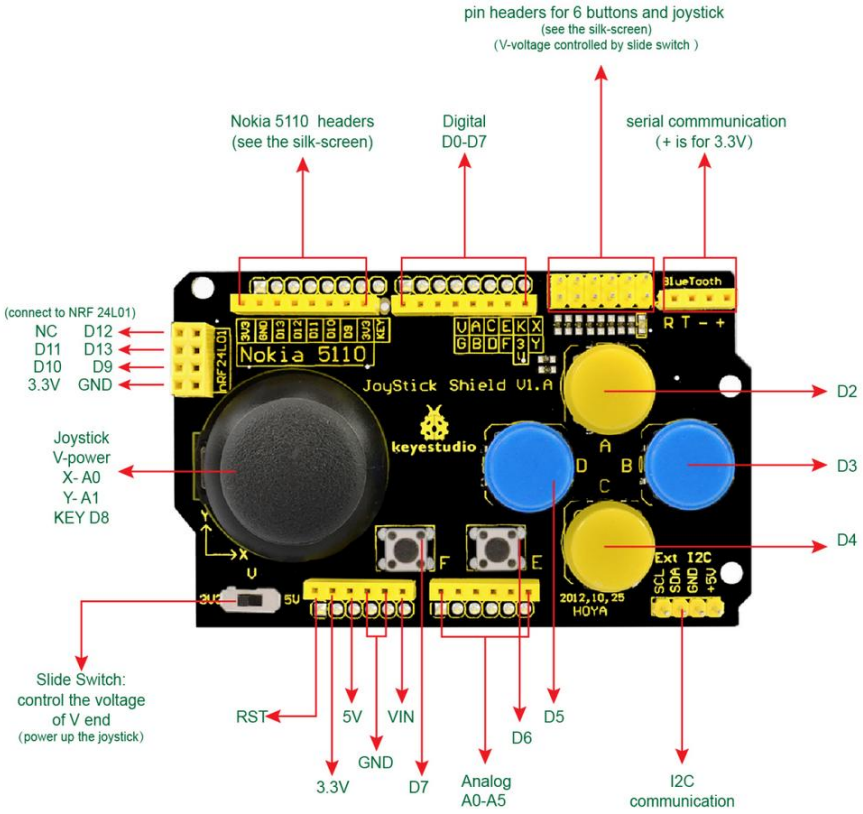
Pin	Name	Connect to MEGA 2560
1	VCC	
2	GND	GND
3	CS	A5
4	RESET	A4
5	DC/RS	A3
6	SDI (MOSI)	51
7	SCK	52
8	LED	A0
9	SDO (MISO)	50
10	T_CLK	46
11	T_CS	45
12	T_DIN	48
13	T_DO	47
14	T_IRQ	49

Item	Symbo	Value			Unit
		Min	Typ	Max	
Supply Voltage for logic	Vcc	2.5	2.8	3.3	V
	Vci	2.5	2.8	3.3	

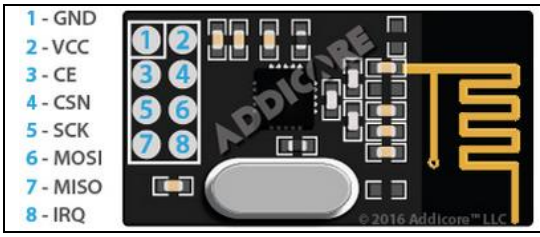
4.21 Joystick Shield V1.A

	Switches have a pull up resistor, when the switch is closed it connects to ground (reading zero). Note: Joystick button is named "Key", found on upper left header rightmost pin.

Information site: https://wiki.keyestudio.com/Ks0153_keyestudio_JoyStick_Shield
https://protosupplies.com/product/funduino-joystick-shield-v1-a/



4.22 Wireless module NFR24L01



Datasheet: C:\Work\Arduino\Documentation\Datasheet_NFR22L01
Vcc: 3.3V (Supply not 5V compatible, Input pins: Vi = max 5.5V).

Total power max = 60mW. @3V3 -> Imax = ~20mA
Transmission: 12mA.
Distance: 100 meter.
125 channels, each channel can have 6 addresses.

Connections to Arduino Nano (Sensor Board):

NFR24L01 pin	Nano pin
3 CE	D7
4 CSN	D8
5 SCK	D13 SCK
6 MOSI	D11 MOSI
7 MISO	D12 MISO
8 IRQ	N.C.

Connections to Arduino Mega (LCD Board):

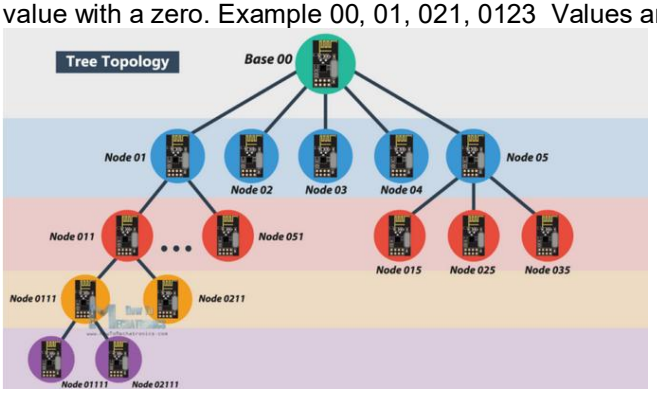
NFR24L01 pin	Mega pin
3 CE	D10
4 CSN	D11
5 SCK	52 SCK
6 MOSI	51 MOSI
7 MISO	50 MISO
8 IRQ	N.C.

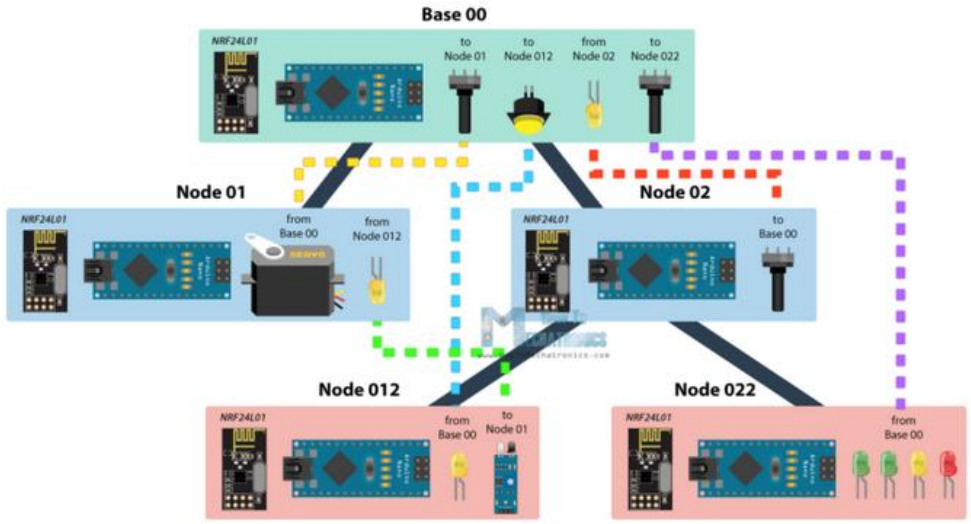
Library: <user>/Documents/Arduino/Libraries/RF24

Tutorial: https:
http://howtomechatronics.com/tutorials/arduino/arduino-wireless-communication-nrf24l01-tutorial/

Network example:
https://howtomechatronics.com/tutorials/arduino/how-to-build-an-arduino-wireless-network-with-multiple-nrf24l01-modules/

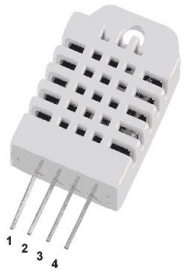
Network notes are defined using octal values. To specify a value as an octal value it is required to prefix the value with a zero. Example 00, 01, 021, 0123 Values are between 0 and 7, value 0181 is incorrect.





4.23 Temperature & Humidity sensor DHT-22 AM2302

DHT22 pins	
1	VCC
2	DATA
3	NC
4	GND



Datasheet: C:\Work\Arduino\Documentation\Datasheet_DHT-22
Vcc: 3.3V – 6V DC.
Pullup of 10k recommended on the Data pin.

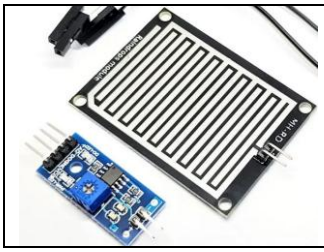
Connections to Arduino Nano:

DTH-22 pin	Nano pin
2 DATA	D4

Library: <user>/Documents/Arduino/Libraries/DHT_sensor_library
<user>/Documents/Arduino/Libraries/Adafruit_Unified_Sensor

This lib requires:

4.24 Rain sensor



Supply: 3.3V – 5V

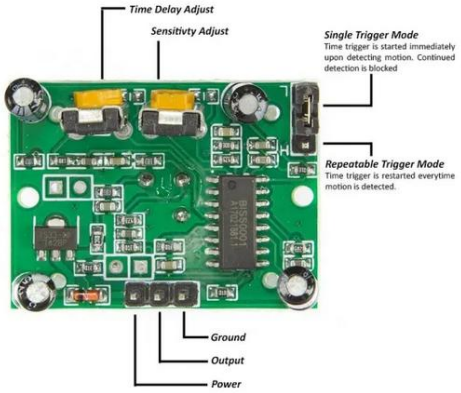
Based on variable resistance. Resistance increase when the sensor is wet and decrease when the sensor is dry. Wet results in lower output voltage. Dry results in higher output voltage

Analog pin can be read by Arduino Ax input pins.
Value = analogRead(A0);

4.25 HC-SR501 PIR sensor



Type: HC-SR501
Supply: 5V



Jumper

L	Output toggles on detected motion
H	Output =1 during detected motion Output =0 after delay time (set by potmeter)

Pins

Power	5V
Output	1 = motion detected
Ground	GND

Test code:

```
int ledPin  = 13;                // LED
int pirPin  = 2;                 // PIR Out pin
int pirStat = 0;                 // PIR status

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(pirPin, INPUT);
  Serial.begin(9600);
}

void loop(){
  pirStat = digitalRead(pirPin);
  if (pirStat) {                          // if motion detected
    digitalWrite(ledPin, HIGH);           // turn LED ON
    Serial.println("Motion detected");
  } else {
    digitalWrite(ledPin, LOW);            // turn LED OFF if we have no motion
  }
}
```

4.26 SR602 micro PIR sensor

A passive infrared (PIR) motion sensor allows you to sense motion, almost always used to detect whether a human has moved in or out of its range.

- Model Number: SR602
- Sensing Distance: Up to 5 meters (0-3.5 meters recommended)
- Output: High Level (H=3.3V/L=0V)
- Power Supply: 3.3V-15V DC
- Quiescent Current: 20uA

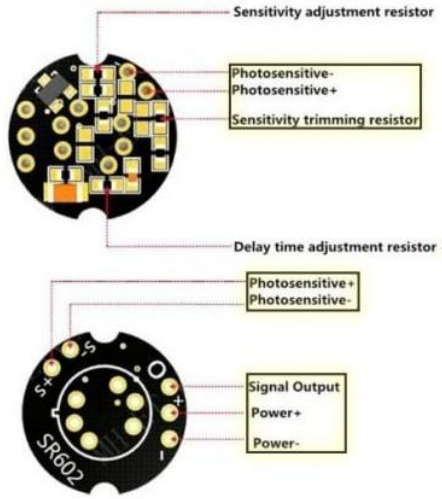
Pinning:

- – : 0V/GND
- +: 3.3V to 15V DC INPUT
- OUT: Logic H/L OUTPUT (3.3V/0V)

1. The high level time of the output of this product is adjustable, 2.5 seconds to 1 hour, the output time set at the factory is 2.5 seconds, and you can change one chip resistor if you need to change it. Please read the resistance value corresponding to the typical delay time.
2. Blocking time, 2 seconds, not adjustable.
3. The factory is defined as repeatable trigger and cannot be changed.
4. The supply voltage of this module is 3.3V to 15V, and the limit voltage is 2.8V to 18V.
5. Module output timing is: output high level for 2 seconds after power-on, then go low, enter standby status. If the delay time is changed, the time when the module outputs a high level after power-on will increase accordingly, which can be understood as the startup time after entering the normal working state after the electric power will increase.
6. After installing the photosensitive element, it does not work during the day, work at night, does not contain photosensitive components, and works all day (There are no photodiodes on the module, please purchase separately if needed. Recommended: ordinary F3 photodiode). It is considered that there is no

photosensitive element installed.

7. This module is very sensitive, pay attention to the installation position, try to avoid the heat source radiation source (air outlet / direct sunlight)
8. The sensitivity of this module is adjustable, and you need to change a chip resistor. See table.
9. When soldering with 220V ordinary soldering iron, it is necessary to unplug the power supply before welding. It is recommended to use 24V low voltage soldering iron, such as domestic white light 936.
10. If you follow the MCU and connect the lamp, please read the reference circuit diagram below.

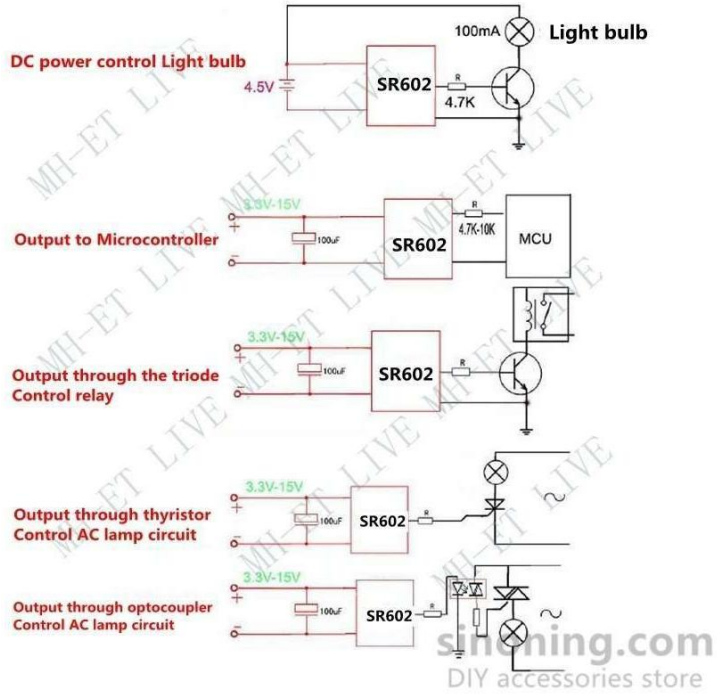


Delay adjustment RL (ohm) (1%)	Delay Time (s)
0 (Default)	2.5
22k	5
39k	8
56k	11
75k	19
91k	35
120k	56
130k	67
154k	130
174k	265
196k	390
221k	522
243k	1050
267k	2095
294k	3150
316M	4200

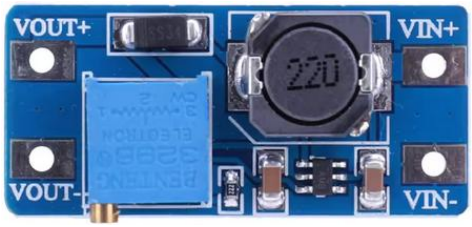




Application examples:



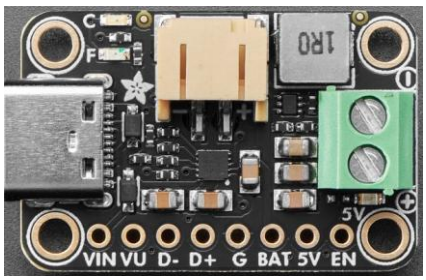
4.27 MT3608 DC-DC BUCK converter/booster 2A



Compact supply module to boost a supply level.
Vin: 2V to 24 V.
Vout: 5V to 28 V. Regulated with potmeter.

Output current:	2.0 A max
Conversion efficiency:	> 93 %
Vripple:	< 100 mV
Switch frequency:	1.2 MHz
Operation temp:	-45 °C to 85 °C
Size:	3,7 x 1,7 x 1,4 cm

4.28 BQ25185 Solar charger + 5V Booster

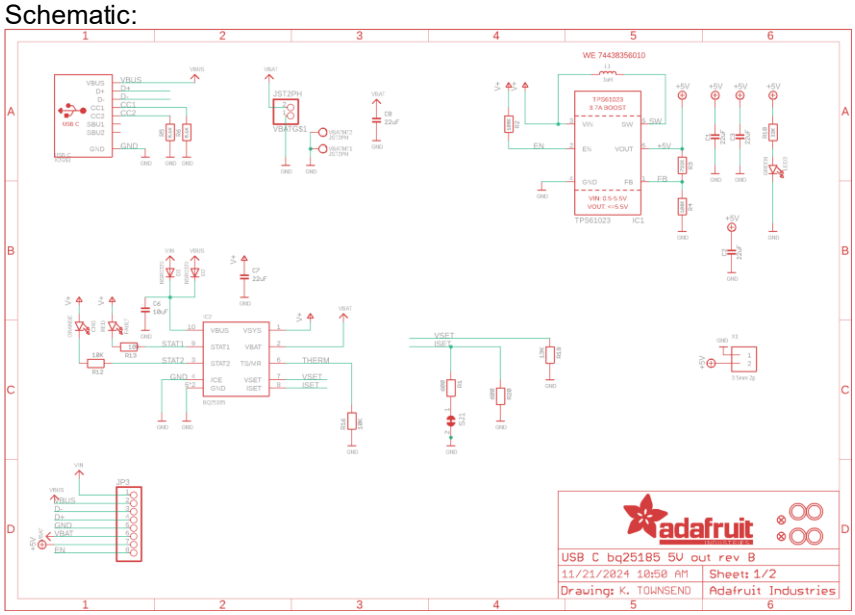


Link: http://adafru.it/6106

- **USB Type C** connector with 5.1k CC resistors so it will work with any computer or power supply to get 5V and up to 1A. Data lines available if needed, on the bottom breakout.
- **Separate DC or Solar Input** - Two pads on the edge can be used to connect a 5 ~ 18V power supply, which can be used instead of USB. If the input is a solar panel, the charging chip will adjust the current draw so that the voltage does not dip below the battery, thus optimizing the solar power input. No large capacitor needed to stabilize it, and you get near-MPPT capability without the cost and complexity of MPPT.
- **Default charge rate of 1A**, but you can cut the jumper on the back to set the rate to 500mA
- **Power Path to Load** - If the 5V load connector is drawing current while the USB / DC/Solar power is attached, it will default to drawing current from the charger and any left over current will go to the battery. That keeps your battery from constantly charging/discharging which will reduce the battery life.
- **Regulated 5V Output** - a boost converter will take the load output from the bq25185 and convert it to 5V at 1 Amp max load. Note the TP61023 cannot turn on/off high loads, it will stall over 200mA. That means its not going to like it if you have a high current load that instantly draws full current on power up, like a resistive dummy load. The vast number of microcontroller projects will start up with <200mA and then a half second later will turn on high current items like LEDs, motors, wifi/radios, or displays so the restriction won't affect it.
- **Three Status LEDs** - Orange **C**harging LED, red **F**ault LED, and Green 3.3V output LED.
- **Enable pad -** to disable the 5V boost converter.

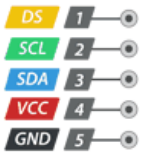
VIN	Solar panel or DC power input. Range: 5V to 18V.
VU	USB connector, power input.
D-	USB connector, negative data line.
D+	USB connector, positive data line.
G	GND, common ground.
BAT	Outputs the battery voltage of the battery connector at the top of the board.
5V	Output voltage
EN	Enable of Buck converter. Open=High(pullup)=Enabled. Pull low to disable.

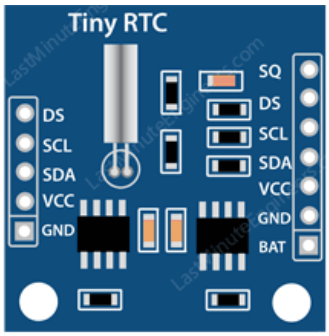
Schematic:

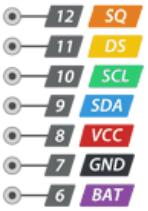


4.29 RTC/EPROM Module DS1307/24C32









VCC = 3.3V to 5V
DS = Temp sensor
SQ = Square Wave

Address EEPROM: 0x50

https://adafruit.github.io/RTClib/html/class_r_t_c___d_s1307.html

eeprom library: https://github.com/cyberp/AT24Cx

Note: Do not use this shield, it is very inaccurate due to a wrong type of crystal soldered on the boards when I bought them. The crystal capacitance does not match with the RTC oscillator requirements which results in an error of several seconds per day.

4.30 RTC/EEPROM Module DS3231/AT24C32



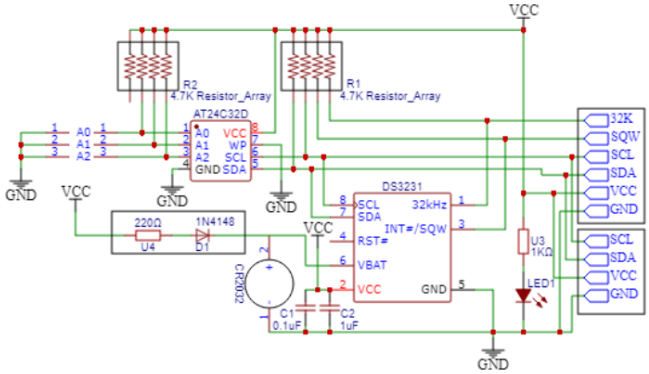
VCC = 3.3V to 5V

Address EEPROM: 0x57 ← the last 3 bits can be controlled by A0, A1, A2 solder pads, default is all '1' for open connections, see schematic which clearly shows this. Thus default of this board is different from DS1307 board!

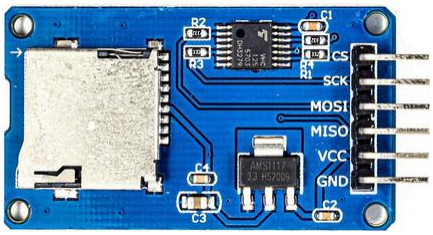
eeprom library: https://github.com/cyberp/AT24Cx

Schematic:
WP=0: normal writes are possible, WP=1 write to the upper quadrant (8/16kbits) is inhibited.

Note: The circuit around the CR2032 is not what I would expect, if Vcc is higher than the battery voltage than Vcc should not charge the battery. This is potentially very dangerous with lithium batteries. It would be better if there was an additions diode in the + line of the battery to block charging. Thus: If a battery is used, then remove the resitor U4 or Diode D1 to disconnect Vcc.

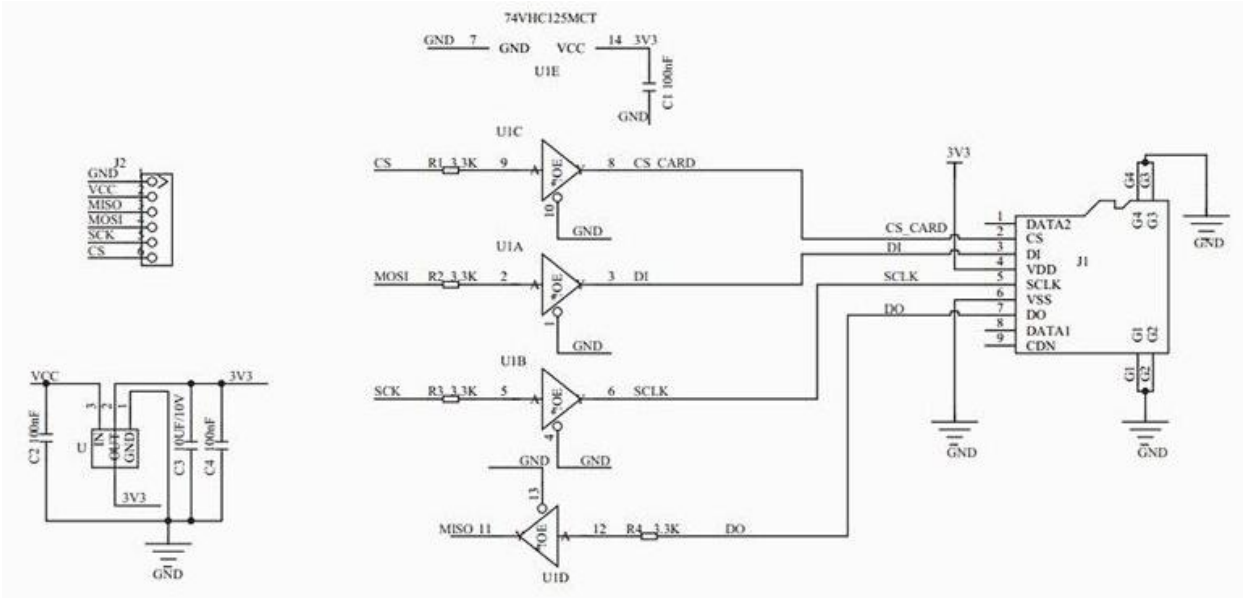


4.31 MicroSD Card adapter

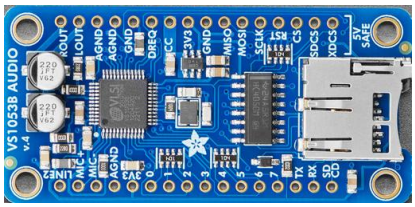


VCC = 3.3V to 5V

Schematic:

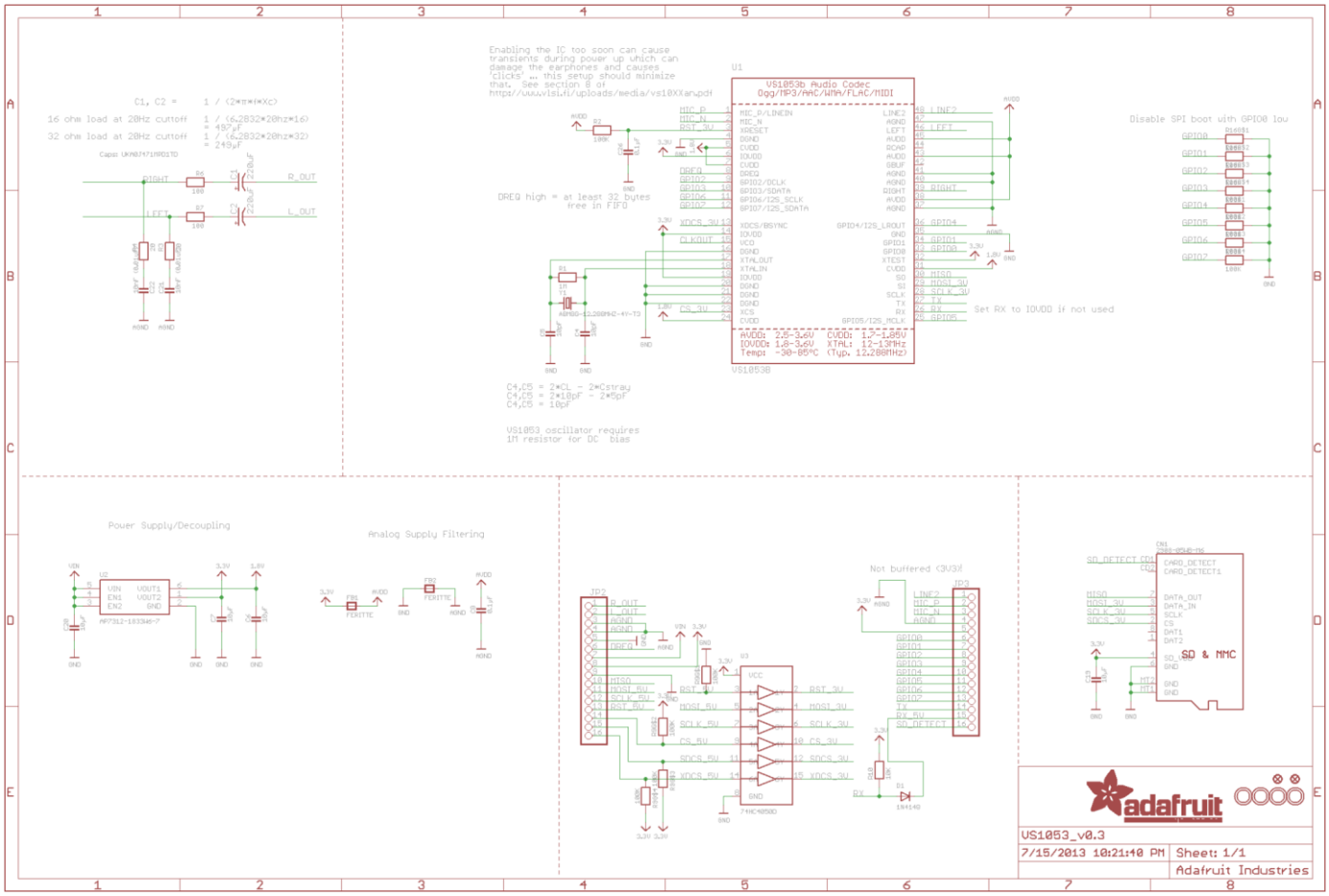


4.32 VS1053 CODEC / MP3 Player



Library for the Adafruit VS1053 Codec Breakout:
Designed specifically to work with the Adafruit VS1053 Codec Breakout ----->
https://www.adafruit.com/products/1381
BSD license, all text above must be included in any redistribution
Stored in: C:\Work\Arduino\Documentation\Datasheet_VS1053_MP3_player

Schematic:



4.33 YX6300 Serial controlled MP3 player



Control interface:	UART TTL interface. A total of four pins (GND, VCC, TX, RX), GND to ground, VCC is the power supply, TX is the TX pin of the MP3 chip, RX is the RX pin of the MP3 chip.
TF card socket:	A micro sd card can be used.
Playbck indicator:	Red light. If it is ready to play or it is paused, it keeps constant on. If playing, it blinks.
Headphone jack:	Can be connected with the headphone or external amplifier.
Mounting holes:	4 screw mounting holes whose diameter is 2.2mm.

1. Support sampling frequency (kHz): 8 / 11.025 / 12 / 16 / 22.05 / 24 / 32 / 44.1 / 48
2. Support file format: MP3 / WAV
3. Support Micro SD card, Micro SDHC Card
4. 30 step adjustable volume
5. UART TTL serial control playback mode, baud rate is 9600bps
6. Power supply can be 3.2 ~ 5.2VDC

See datasheet for serial operation (commands).
On the board the YX6300 device is mounted. Other boards (YX5300, als often have this newer chip).

The commands are listed in a demo program.
A normal starup is:

```
19:00:17.746 -> Command: CMD_SEL_DEV
19:00:17.778 -> Sending: 0X7e 0Xff 0X06 0X09 0X00 0X00 0X02 0Xef
19:00:18.260 -> 0X7e 0Xff 0X06 0X3f 0X00 0X00 0X02 0Xfe 0Xba 0Xef  -> ACK
19:00:18.323 -> 0X7e 0Xff 0X06 0X90 0X00 0X00 0X02 0Xfe 0X69 0Xef  -> Card inserted or removed
```

A start of a specified file in a specifed folder, including return value when done:

```
19:26:24.467 -> Command: CMD_PLAY_FOLDER_FILE
19:26:24.499 -> Sending: 0X7e 0Xff 0X06 0X0f 0X00 0X04 0X01 0Xef
19:26:25.080 -> 0X7e 0Xff 0X06 0X3d 0X00 0X00 0X10 0Xfe 0Xae 0Xef  -> Completed play num 16
19:26:25.177 -> 0X7e 0Xff 0X06 0X3d 0X00 0X00 0X10 0Xfe 0Xae 0Xef  -> Completed play num 16
```

Note: It is unclear why the reply is seen twice….

4.34 PAM8610 / HW-210 Stereo audio amplfier 2 x 10W Class-D 12V



About this item

- 2.0HIFI double channel with each channel output 10W; high efficiency of 90%
- It uses PAM8610 chip, 10W (per channel) D-level stereo audio amplifier
- With over-current/over-temperature/short circuit protection
- Low THD + N (0.1%) without filter; small static current; energy efficient; small heating generator
- Low EMI; with power off/noise reduction function; low noise generator

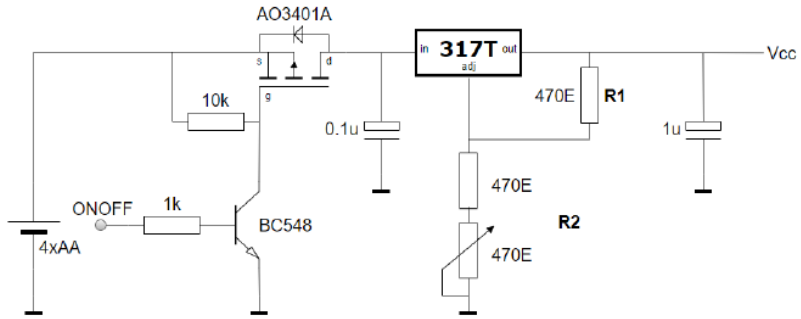
4.35 Low power mode circuit (On / Off)

This circuit uses a P channel MOSFET to switch on/off the main supply. The idea is that the CPU operates in sleep mode, and when activated, it turns on the main supply. For this purpose a GPIO pin of the CPU is connected tot he ONOFF input of this circuit. Logic 0 results in power off, Logic 1 results in power on.

The ONOFF is connected to a transistor and not the gate of the MOSFET directly because this gate could have the same voltage as the Source of the MOSFET, which is too high for a GPIO pin and would damage that pin.

The Drain euqals Vbatt when the gate is zero (pulled low). When ONOFF equals logic 1, the transistor conducts, leading to ~0V on the collector. Thus ONOFF=1 results in power on.

MOSFET's which can be used:
AO3401A V_{DS} = -30V
IRLML6401 V_{DS} = -12V



Vo = 1.25*(1 + R2/R1) + Iadj*R2
Term "Iadj*R2" is not significant
Then: R2 = R1 (Vo/1.25 - 1)
Choose: R1 = 470, Vo = 3V3 => R2 = 771

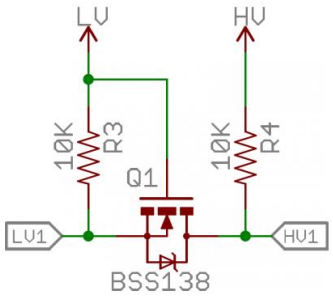
Assume: R2min = 470, R2max = 940Vo_min = 2.5V, Vo_max = 3.75V

Assume: R2min = 470, R2max = 1470Vo_min = 2.5V, Vo_max = 5.16V

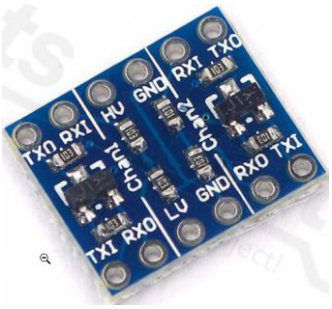
4.36 Level shifter circuit 1

https://learn.sparkfun.com/tutorials/bi-directional-logic-level-converter-hookup-guide/all

The circuit uses a single N-channel MOSFET and two pull-up resistors to realize bi-directional level shifting.



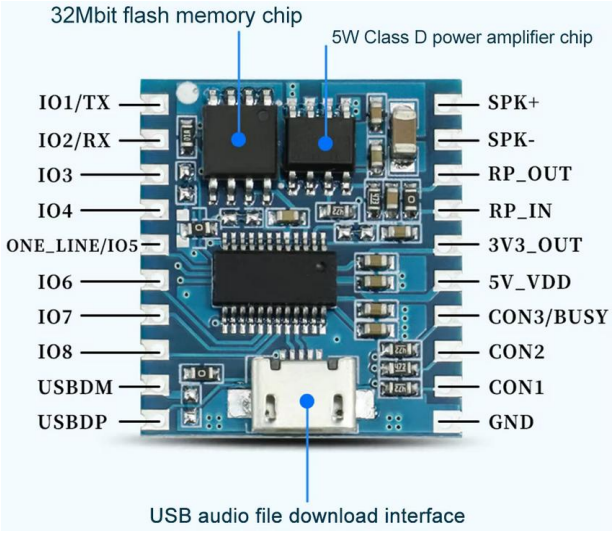
4.37 Level shifter circuit 2





U(RX_LV) = 2/3 U(RX_HV), U(RX_HV) = 5.0V ➔ U(RX_LV) = 3.3V

4.38 DY-SV19R Speech Module MP3/WAV



Support MP3 and WAV
Output: 24 bit DAC
Dynamic range: 90 dB
SNR: 85 dB
Flash: 32Mbit (4Mbyte), update via USB
Config: 3 pins to support 7 working modes
Supply: 3.7-5.0V
Amplifier: Class D, 5 W, 4-8 ohm speaker
Support: External potentio meter

Chips:
Control: DY1900A
Amplifier: LTK51280

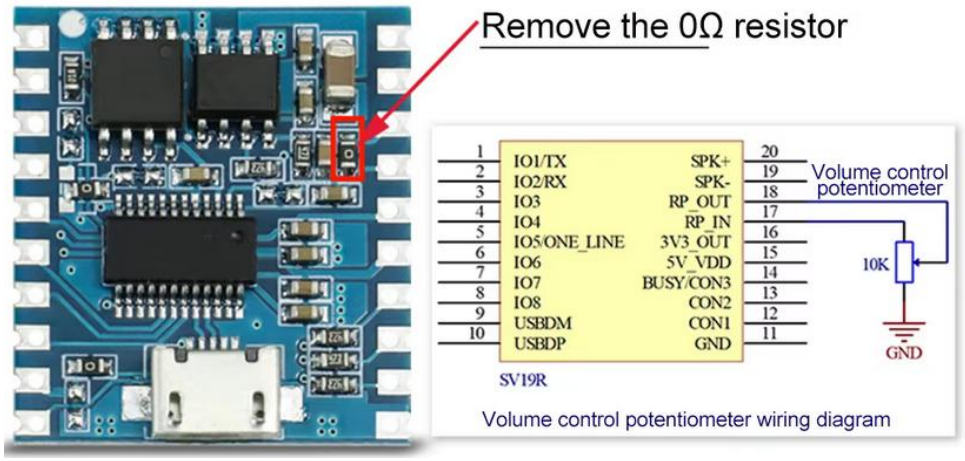
https://grobotronics.com/images/companies/1/datasheets/DY-SV5W%20Voice%20Playback%20ModuleDatasheet.pdf?1559812879320&srsltid=AfmBOordXmo9ee3_odCjw58c-apptlA6cngEpEHn4ok7puSMgjPfOdfk

Note:
This module supports several modes of operation. For the UART mode I found out that playing a file named 00010.mp3 by using (10) as a parameter to identify this file does NOT work if only this file is on the flash available. It works if there are 10 files available. Example: If there are 6 files available and the sixth file is 00010.mp3 then the nr for the command (play specified) to play this file is 6 and not 10, this is tested where file 00006.mp3 did also not exist on flash, it plays the sixth file.

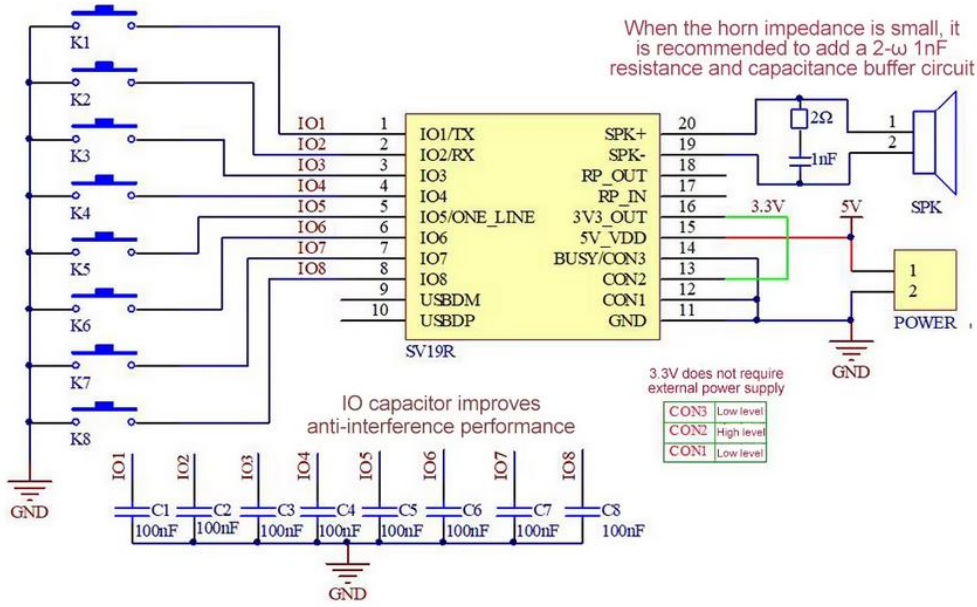
Operating modes:

Trigger mode	Configure the pin Settings			Trigger description
	CON3	CON2	CON1	
IO independent mode 0	0	1	0	IO and GND are short-connected (low level) to trigger playing. After disconnection (high level of IO), the triggered track is played and stopped. If IO and GND keep the short-connected state, the triggered track will be played all the time. One on one triggers 8 tracks.
IO independent mode 1	0	1	1	IO and GND are short-connected (low level) to trigger playing, and immediately stop playing after disconnection (high level of IO). If I0 and GND keep the short-connected state, the triggered tracks will be played all the time. One on one triggers 8 tracks.
IO independent mode 0	0	0	0	IO combination triggers playback, and stops after release (all 10 is high level) after playing the triggered tracks. If combination 10 keeps the triggered state, it keeps playing the triggered tracks: 255 tracks are triggered by MCU control.
IO independent mode 1	0	0	1	Combination 10 triggers playing, and immediately stops playing after release (all 10 are high). If combination 10 keeps the triggered state, the triggered track will be played continuously. MCU control triggers 255 first song days
UART Serial port mode	1	0	0	MCU sends control commands to voice module through UART to control playback, music selection, interstitial and other functions. See UART mode application manual for details
Single bus mode One line				MCU sends instructions through the single-line private protocol to control module playback, music selection, volume adjustment and other functions. See the application manual of single-line serial port mode for details
Standard MP3 mode	1	0	1	By default, it plays automatically during power-on. The external button can adjust the volume, up and down bend, play/pause, EQ and other functions

Volume control by external potentio meter:

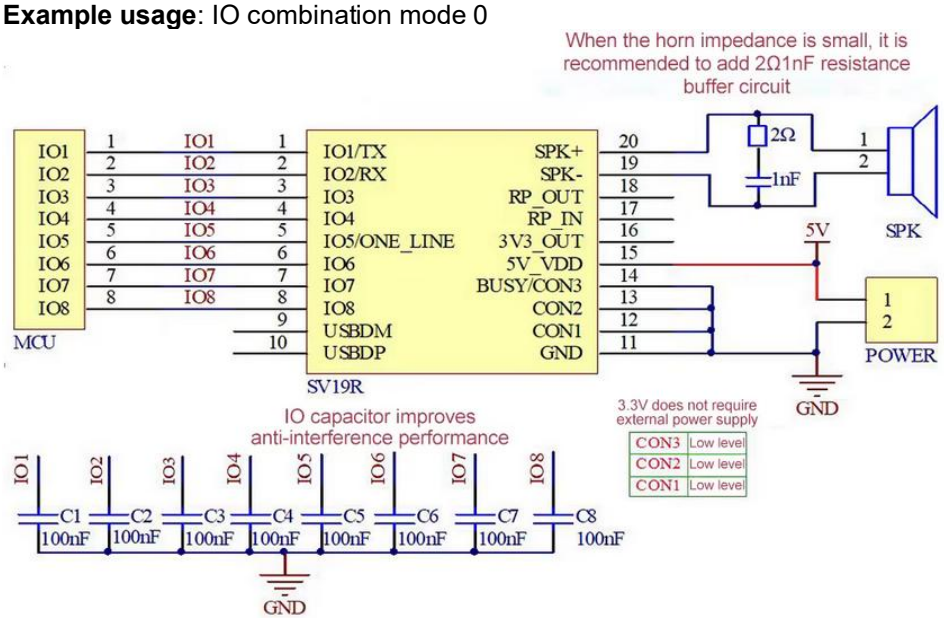


Example usage: IO independent mode



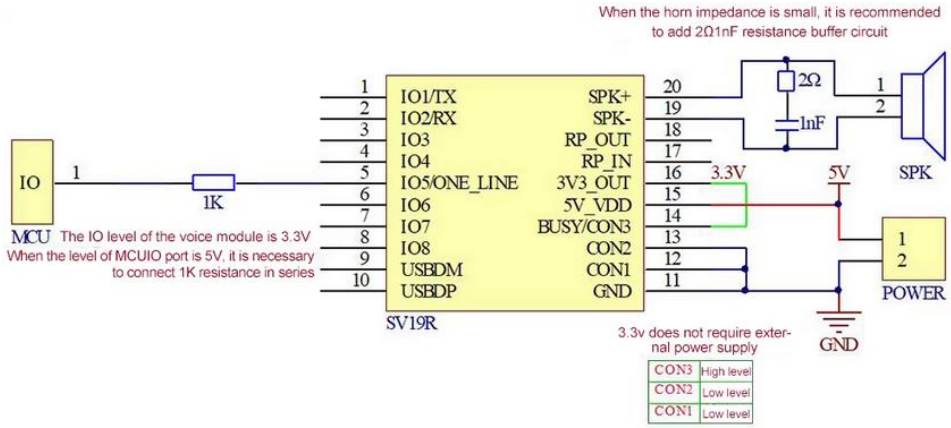
IO status	Play track
IO1 == GND	00001.mp3
IO2 == GND	00002.mp3
IO3 == GND	00003.mp3
IO4 == GND	00004.mp3
IO5 == GND	00005.mp3
IO6 == GND	00006.mp3
IO7 == GND	00007.mp3
IO8 == GND	00008.mp3

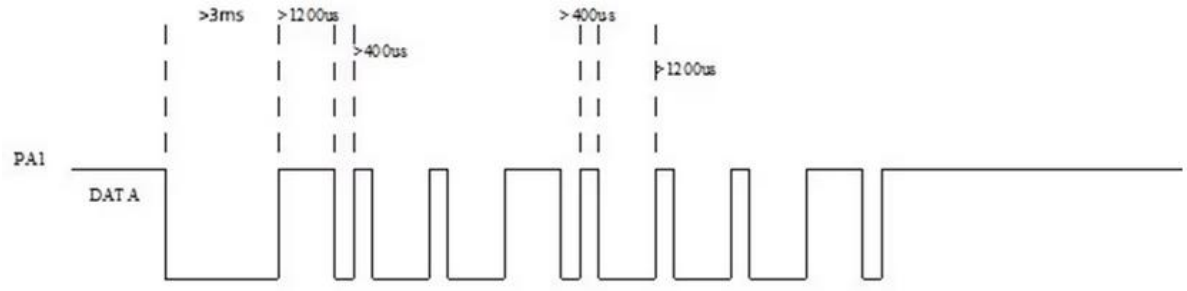
Example usage: IO combination mode 0

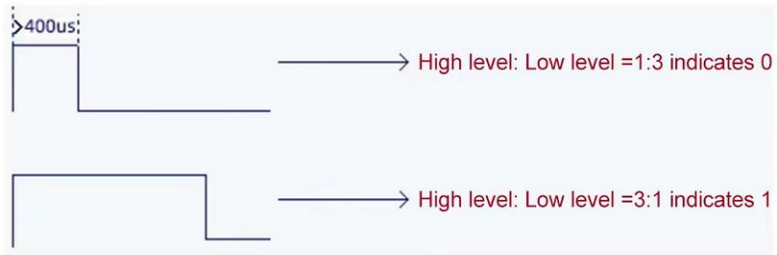


IO8	IO7	IO6	IO5	IO4	IO3	IO2	IO1	Play track
1	1	1	1	1	1	1	0	00001.mp3
1	1	1	1	1	1	0	1	00002.mp3
1	1	1	1	1	1	0	0	00003.mp3
…	…	…	…	…	…	…	…	…
0	0	0	0	0	0	0	0	00255.mp3

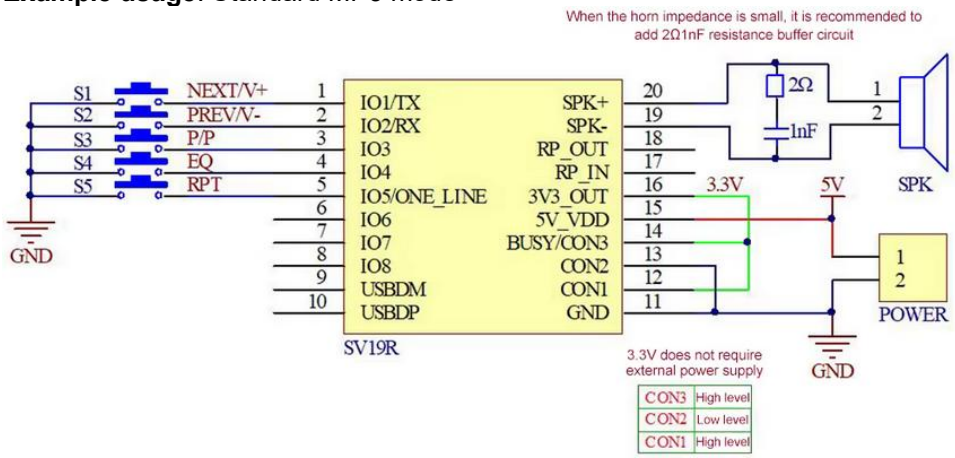
Example usage: Single-bus module ONE_LINE







Example usage: Standard MP3 mode



Datasheet info of the control chip:

Work Mode Configuration											
Control Mode	Configuration Pin			I/O Function							
	CON3	CON2	CON1	IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0
I/O Integrated Mode 0	0	0	0	Key combination play, can play 2^8-1(255) Songs.							
I/O Integrated Mode 1	0	0	1	Level combination play, can play 2^8-1(255) Songs.							
I/O Independent Mode 0	0	1	0	Song8	Song7	Song6	Song5	Song4	Song3	Song2	Song1
I/O Independent Mode 1	0	1	1	Song8	Song7	Song6	Song5	Song4	Song3	Song2	Song1
UART Mode	1	0	0							RXD	TXD
One-Line Mode	1	0	0				TXD				
Standard MP3 Mode	1	0	1				RPT	EQ	P/P/MODE	PREV/V-	NEXT/V+
Note:											
1>. "key combination play" : Return to the original high level after the corresponding level from I/O0-I/O7 output, similar to the key triggered once.Similar instantaneous switch.											
2>. "Level combination play" :The trigger signal remains the same, similar to a self-locking switch.											
3>.The difference between "I/O Integrated/Independent Mode 0" and "I/O Integrated/Independent Mode 1" :Mode 0 will continue playing the current song to the end after release level .Mode 1 will stop playing immediately after release level.											

I/O Integrated Mode 0 (Key combination playing).									
Note: the song must be named for 5bit.									
IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0	Song	It will stop playing current song to the end after I/O0-7 release input signal (return to high) at 'I/O Integrated Mode 0'. It will playing new song when get new input signal during playing and stop after end of song. It will play repeatedly if keep input. Busy pin will output valid signal(High) during playing.Music control as following:
1	1	1	1	1	1	1	0	00001.mp3	
1	1	1	1	1	1	0	1	00002.mp3	
1	1	1	1	1	1	0	0	00003.mp3	
1	1	1	1	1	0	1	1	00004.mp3	
1	1	1	1	1	0	1	0	00005.mp3	
1	1	1	1	1	0	0	1	00006.mp3	
1	1	1	1	1	0	0	0	00007.mp3	
.....	.....	.....	.....	.....	.....	.....	.....	.....	
0	0	0	0	0	0	0	0	00255.mp3	

I/O Integrated Mode 1 (Level combination playing)									
IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0	Song	It will keep playing current song when get trigger signal.It will stop playing immediately after release level.Busy pin will output valid signal(High) during playing.
1	1	1	1	1	1	1	0	00001.mp3	
1	1	1	1	1	1	0	1	00002.mp3	
1	1	1	1	1	1	0	0	00003.mp3	
1	1	1	1	1	0	1	1	00004.mp3	
1	1	1	1	1	0	1	0	00005.mp3	
1	1	1	1	1	0	0	1	00006.mp3	
1	1	1	1	1	0	0	0	00007.mp3	
.....	.....	.....	.....	.....	.....	.....	.....	.....	
0	0	0	0	0	0	0	0	00255.mp3	

I/O Independent Mode 0 (Key independent controlling)									
IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0	Song	I/O0-I/O7 independently controls 8 songs.It will stop playing current song to the end after I/O0-7 release input signal(return to high);It will playing new song when get new input signal during playing and stop after end of song;It will play repeatedly if keep input;Busy pin will output valid signal(High) during playing.
1	1	1	1	1	1	1	0	00001.mp3	
1	1	1	1	1	1	0	1	00002.mp3	
1	1	1	1	1	0	1	1	00003.mp3	
1	1	1	1	0	1	1	1	00004.mp3	
1	1	1	0	1	1	1	1	00005.mp3	
1	1	0	1	1	1	1	1	00006.mp3	
1	0	1	1	1	1	1	1	00007.mp3	
0	1	1	1	1	1	1	1	00008.mp3	

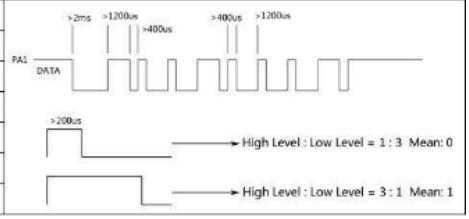
I/O Independent Mode 1 (Level independent controlling)									
IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0	Song	I/O0-I/O7 independently controls 8 songs.It will keep play repeatedly specify the triggered song.It will stop playing immediately after release level.Busy pin will output valid signal(High) during playing.
1	1	1	1	1	1	1	0	00001.mp3	
1	1	1	1	1	1	0	1	00002.mp3	
1	1	1	1	1	0	1	1	00003.mp3	
1	1	1	1	0	1	1	1	00004.mp3	
1	1	1	0	1	1	1	1	00005.mp3	
1	1	0	1	1	1	1	1	00006.mp3	
1	0	1	1	1	1	1	1	00007.mp3	
0	1	1	1	1	1	1	1	00008.mp3	

UART Mode					
Communication Format					
Adopt full duplex serial port communication. Baud rate 9600, data bits 8, stop bit 1, check bit N.					
Start Code	Command Type	Data Length (n)	Data 1	Data n	Check Bit (SM)
Command Code: fixed to 0xAA.					
Command Type: used to distinguish the type of command.					
Data Length: the number of bytes of data in an command.					
Data: Relevant data in command, when length of data is 1, means there is only CMD and no data bits.					
Check Bit: Low 8 bits of sum of all bytes. that is, When start code and data are added, take out low 8 bits.					
Data format: Sent data or command, high 8-bit data is in front, low 8-bit is in the back.					
Communication Protocol					
The following is a data definition for the return and identification of the chip.					
A. Playing State definition: the system is on the stop state when power on.					
	00(stop)	01(play)	02(pause)		
B. Disk character definition: it is stopped after the switch disk.					
	USB:00	SD:01	FLASH:02	NO_DEVICE: FF	
C. Volume: the volume is 31grades, 0-30.The default is 20grade.					
D. Play mode: the default is the single stop when power on.					
Cycle for all songs (00) : play the whole songs in sequence and play it after the play.					
Single cycle (01) : play the current song all the time.					
Single stop (02) : Only play current song once and then stop.					
Random play (03) : random play.					
Directory loop (04) :Play in current folder in order, then play by play.Directory don't contain subdirectory.					
Directory random (05): random play in the current folder, and directory does not contain subdirectory.					
Directory order play(06):Play current folder in order & stop after play.Directory not include subdirectory.					
Sequential play (07) : play the whole songs in order and stop after it is played.					
E. EQ definition: the default EQ is NORMAL(00).					
	NORMAL(00)	POP(01)	ROCK(02)	JAZZ(03)	CLASSIC(04)
F. Composition play definition: combination play is combined by filename. The file requirements are stored under the "XY" file. You can change the name of the file you want to combine to two bytes, which is generally recommended as a number. Such as: 01. Mp3, 02. Mp3.					

UART Communication Command		
Control Command		
Command	Command code	Return
Play	AA 02 00 AC	None
Pause	AA 03 00 AD	None
Stop	AA 04 00 AE	None
Previous	AA 05 00 AF	None
Next	AA 06 00 B0	None
Volume +	AA 14 00 BE	None
Volume -	AA 15 00 BF	None
Previous file	AA 0E 00 B8	None
Next file	AA 0F 00 B9	None
Stop playing	AA 10 00 BA	None
Query Command		
Command	Command Code	Return
Query play status	AA 01 00 AB	AA 01 01, play status, SM
Query current online drive	AA 09 00 B3	AA 09 01, drive, SM
Query current play drive	AA 0A 00 B4	AA 0A 01, drive, SM
Query Number of songs	AA 0C 00 B6	AA 0C 02S.N.H S.N.L SM
Query current song	AA 0D 00 B7	AA 0D 02 S.N.H S.N.L SM
Query folder directory song	AA 11 00 BB	AA 11 02 S.N.H S.N.L SM
Query folder Number of songs	AA 12 00 BC	AA 12 02 S.N.H S.N.L SM

UART Communication Command					
Control Command			Query Command		
Command	Command Code	Return	Command	Command code	Return
Play	AA 02 00 AC	None	Query play status	AA 01 00 AB	AA 01 01, play status, SM
Pause	AA 03 00 AD	None	Query current online drive	AA 09 00 B3	AA 09 01, drive, SM
Stop	AA 04 00 AE	None	Query current play drive	AA 0A 00 B4	AA 0A 01, drive, SM
Previous	AA 05 00 AF	None	Query Number of songs	AA 0C 00 B6	AA 0C 02S.N.H S.N.L SM
Next	AA 06 00 B0	None	Query current song	AA 0D 00 B7	AA 0D 02 S.N.H S.N.L SM
Volume +	AA 14 00 BE	None	Query folder directory song	AA 11 00 BB	AA 11 02 S.N.H S.N.L SM
Volume -	AA 15 00 BF	None	Query folder Number of songs	AA 12 00 BC	AA 12 02 S.N.H S.N.L SM
Previous file	AA 0E 00 B8	None			
Next file	AA 0F 00 B9	None			
Stop playing	AA 10 00 BA	None			

Setting Command			
Command	Command code	Return	Remark
Set Volume	AA 13 01 VOL SM	None	VOL:0x00-0xFF
Set Loop mode	AA 18 01 Loop-mode SM	None	Loop-mode:0x00-0x07
Set Cycle times	AA 19 02 H L SM	None	H:0x00-0xFF L:0x00-0xFF
Set EQ	AA 1A 01 EQ SM	None	EQ:0x00-0x04
Specified Song	AA 07 02 S.N.H S.N.LSM	None	S.N.H:0x00-0xFF S.N.L:0x00-0xFF
Specified Path	AA 08 Length Drive Path SM	None	Length:0x00-0xFF
			Drive:0x00-0xFF
			Path:0x00-0xFF
Switch Specified Drive	AA 0B 01 Drive SM	None	Drive:0x00-0xFF
Specified song to be interplay	AA 16 03 Drive S.N.H S.N.L SM	None	Drive:0x00-0xFF
			S.N.H:0x00-0xFF
			S.N.L:0x00-0xFF
Specified path to be interplay	AA 17 Length Drive Path SM	None	Length:0x00-0xFF
			Drive:0x00-0xFF
			Path:0x00-0xFF
Select but no play	AA 1F 02 S.N.H S.N.L SM	None	S.N.H:0x00-0xFF S.N.L:0x00-0xFF

One_line Single Bus Mode		
Command(HEX)	Function	Note
0x00	No. 0	The number 0-9 has corresponding functions, such as selecting music, setting the volume, setting EQ, setting cycle mode, setting channel, setting the repertoire, and sending the digital at first and then send function command.
0x01	No. 1	
0x02	No. 2	
0x03	No. 3	
0x04	No. 4	
0x05	No. 5	
0x06	No. 6	
0x07	No. 7	
0x08	No. 8	
0x09	No. 9	
0x0A	Number reset	Sent the number of Cleared
0x0B	Confirm choosing song	Cooperate with Numbers to achieve.
0x0C	Volume setting	
0x0D	EQ setting	
0x0E	Loop mode setting	
0x0F	Channel setting	
0x10	Interplay song setting	
0x11	Play	Note: "selection" and "interplay" are played according to the track name, for example, the track is named "00123. Mp3", and the selected data is "0x01", "0x02" "0x03" "0x0B", and the selection is completed.
0x12	Pause	
0x13	Stop	
0x14	Previous	
0x15	Previous directory	
0x16	Next directory	
0x17	SD card selection	
0x18	SD card selection	
0x19	U disk selection	
0x1A	FLASH selection	
0x1B	System sleep	
0x1C	Stop Playing	

4.39 TF Card MP3 Audio Decoder Board (2W, 5V)

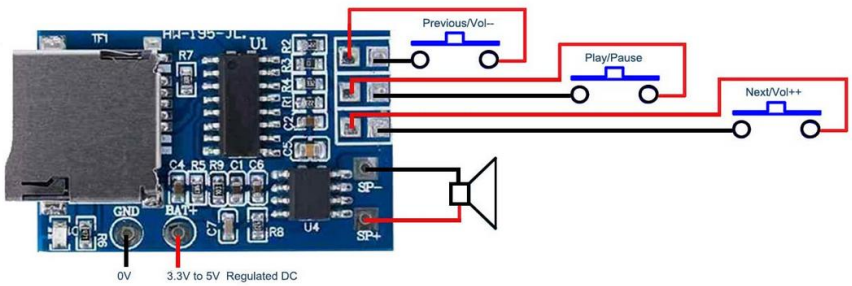


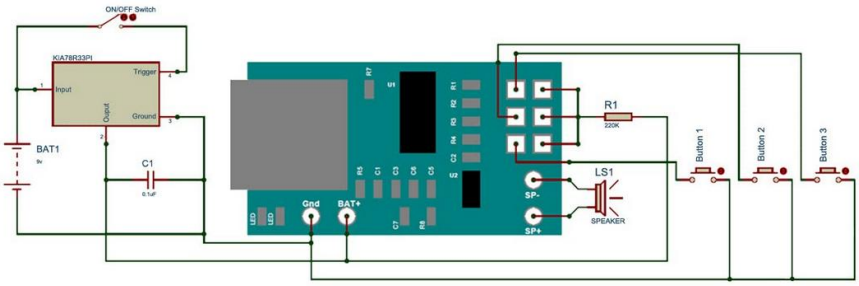
Type: GPD2846A (clone ?)
IC: AC24BP20794

Supply: 3.7 – 5.0V
Format: MP3
Max TF card size: 16GB
Amplifier: 3W (@Vcc=4.5V)

BAT: Supply Vcc
GND: Supply GND
SP: Audio + output
SP-: Audio – output

The 8pin IC is the amplifier, add heatsink when needed
IC: 8002A





Link:
https://forum.arduino.cc/t/gpd2846a-tf-card-mp3-decoder-board-arduino/653684

5 Coding notes

5.1 Installation of libraries

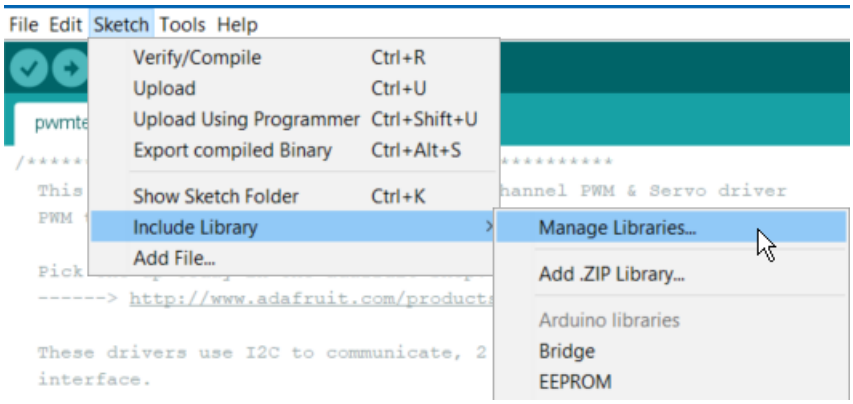
Location of the library:
C:\Users\<USERNAME>\Documents\Arduino\libraries

5.1.1 Library installation from the GUI: Manage Libraries

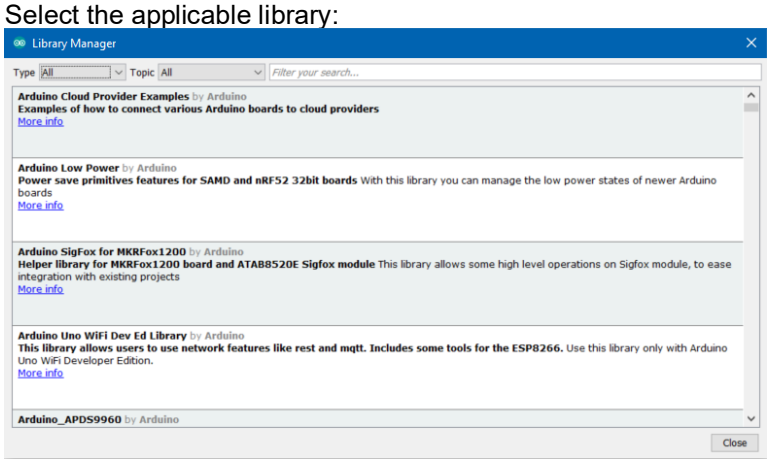
When the GUI is used to install libraries, then they are automatically recognized during compiling, meaning they are found on the expected location.

Info: https://www.arduino.cc/en/Guide/Libraries

Use Sketch -> Include Library -> Manage Libraries to install a library from a list of known available libraries on the internet.



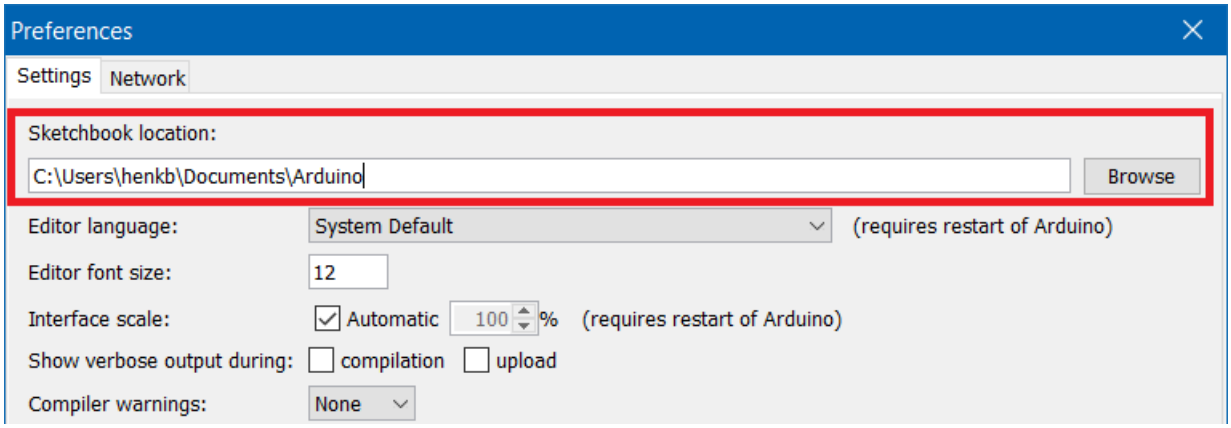
Select the applicable library:



In case of the Adafruit 12 channel PWM servo driver:

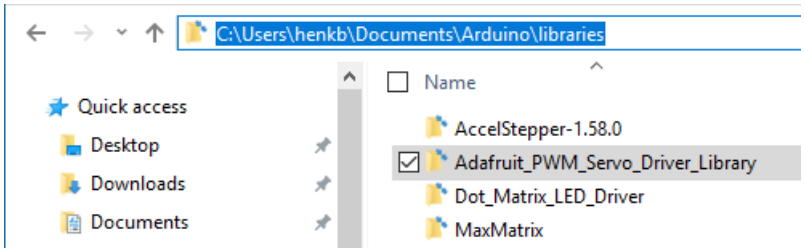


Location of the library:
The location is defined by the IDE, but not directly identifiable as the library location:
File->Preferences: Sketchbook location:



5.1.2 Library installation: Manual (.zip file)

You can also manually install libraries in this directory, by copying the unzipped root folder into the library.
When downloaded from Github the directory name is <library>-master. Remove the “-master” part.
Check: c:\Users\henkb\Documents\Arduino\libraries



5.2 Variable types

				Range		
Type	Sign	Bytes	Bits	Min	Max	Other Info.
char	signed	1	8	-128	127	ASCII
char	unsigned	1	8	0	255	ASCII
byte		1	8	0	255	
int (Uno +)	signed	2	16	-32768	32767	Uno model +others.
short		2	16	-32768	32767	
int (Uno +)	unsigned	2	16	0	65535	Uno model +others.
word		2	16	0	65535	Same as unsigned int.
int (Due)	signed	4	32	-2147483648	2147483647	Due model only.
long	signed	4	32	-2147483648	2147483647	Append with 'L'.
int (Due)	unsigned	4	32	0	4294967295	Due model only.
long	unsigned	4	32	0	4294967295	
float		4	32	-3.4028235E+38	3.4028235E+38	6-7 dec digits of precision.
double (Uno +)		4	32	-3.4028235E+38	3.4028235E+38	Same as float.
double (Due)		8	64	(small)	(BIG)	Double precision float.

The types like uint16_t tell exactly what it is.
The type byte is an other notation for uint8_t.

5.3 Commands

delay(x)	Delay x [ms] (works only when interrupts are not disabled)
delayMicroseconds(x)	Delay x [us]
Serial.begin(115200);	Set baudrate of serial interface
Serial.write(value);	Value sent as single byte
Serial.write(string);	String sent as a series of bytes
Serial.write(buf, len);	Array sent as series of byes until len is reached
Serial.println(item);	Item can by of any time
Serial.println(item, format);	Item is a value, format: DEC, HEX, OCT, BIN

Ports Digital:



For digital ports use the number in the pink box (14 in example above) as the port number.

```
digitalRead(port);
digitalWrite(port, value);
```

Ports Analog:



These are identified by the ADC number, 0=ADC0, which is on pin A0 of the Nano.

```
#define Pot_Left_UD       A1                  // Analog input of pot meter on pin A1
int val;
val = analogRead(Pot_Left_UD);                // Returns value between 0 and 1023 (10 bit value)
```

Open Serial Monitor Ctrl+Shift+m

5.4 Interrupts

5.4.1 Timer generating an interrupt every N milliseconds

Compare_Match_REG = ($16*10^6$ / (Prescaler_Value * Interrupt_Freqency)) - 1

Interrupt time = 500 ms => Interrupt frequency is 2 Hz.
Prescaler_Value = 1024

Compare_Match_REG = 7811
Timer 1 is required to store this value (16 bits), timer 0 and timer 2 are 8 bits.

in setup():

```
  cli();           // Stop interrupts

  // Set timer1 interrupt at 2Hz
  TCCR1A = 0;       // Set entire TCCR1A register to 0
  TCCR1B = 0;       // Same for TCCR1B
  TCNT1  = 0;       // Initialize counter value to 0
  // Set compare match register for 1hz increments
  OCR1A = 7811;     // (16*10^6) / (1024*2) - 1 (must be <65536)
  // Turn on CTC mode
  TCCR1B |= (1 << WGM12);
  // Set CS10 and CS12 bits for 1024 prescaler
  TCCR1B |= (1 << CS12) | (1 << CS10);
  // Enable timer compare interrupt
  TIMSK1 |= (1 << OCIE1A);

  sei();           // Allow interrupts
```

The commands you want to execute during these timer interrupts are located in the Arduino sketch encapsulated in the following:

```
ISR(TIMER0_COMPA_vect) {  // Change the 0 to 1 for timer1 and 2 for timer2
    //Interrupt commands here
    //Note: global variables which are changed should be declared as volatile
}
```

5.4.2 Interrupt from an input pin

Not all pins are interrupt pins. On the Nano, pin 2 and 3 are interrupt pins.

```
const byte Sensor_IR_Left = 3;

volatile bool obstacle_left = false;

setup {
  pinMode(Sensor_IR_Left, INPUT);
  attachInterrupt(digitalPinToInterrupt(Sensor_IR_Left), Obstacle_Left_Found, LOW);
}

void Obstacle_Left_Found() {
  obstacle_left = true;
}

void loop() {
  while(!obstacle_left) {
    // do something
  }
}
```

Function digitalPinToInterrupt(pin_number), returns the interrupt number, or returns -1 if this pin is not an interrupt pin.

5.5 Graphical libraries

5.5.1 Adafruit GFX library

Documentation: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-gfx-graphics-library.pdf
Works ok with ESP32C3 Super Mini combined with MSP3520 / ILI9488

One must use this v1.4.7 version, later versions creat an error message
Error message: fatal error: esp32-hal-periman.h: No such file or directory
This is prevented by using v1.4.7:

```
lib_deps =
  moononournation/GFX Library for Arduino@1.4.7
```

Adding this dependency in the platformio.ini file also installs this library when it is not yet available.

Development of code can be done in the Arduino IDE, however this has a huge disadvantage: Each edit results in a complete re-compile of all code, thus including the 'static' library. This takes on a desktop computer with 8 CPU's minutes. This makes it totally unrealistic to develop code in this IDE.

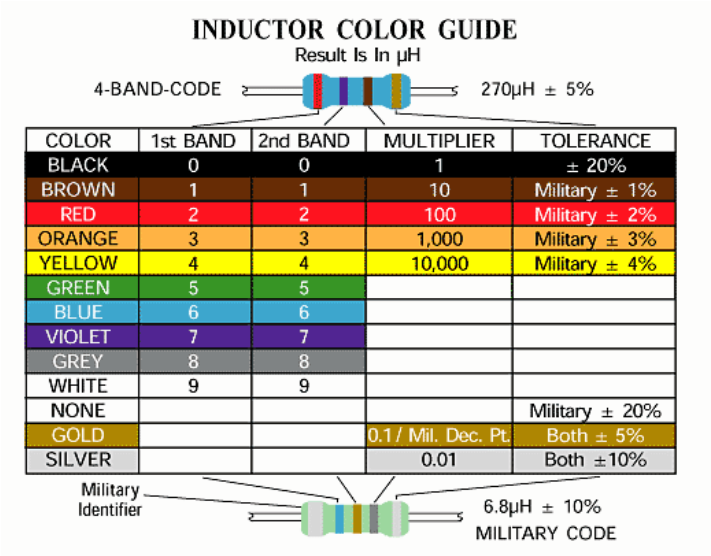
Development of code can be done in Visual Studio Code in combination with PlatformIO. This IDE is lightyears ahead of the Arduino IDE and a re-compile of the same project takes now instead of many minutes a couple of seconds (tested on the same desktop)

5.5.2 TFT eSPI

Does NOT work with ESP32C3 Super Mini combined with MSP3520 / ILI9488
The library crashes in the init routine, it also states that this combination is not tested.

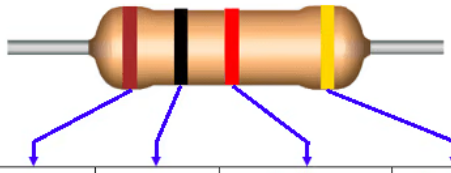
6 Color coding guides

6.1 Inductors



COLOR	1st BAND	2nd BAND	MULTIPLIER	TOLERANCE
BLACK	0	0	1	± 20%
BROWN	1	1	10	Military ± 1%
RED	2	2	100	Military ± 2%
ORANGE	3	3	1,000	Military ± 3%
YELLOW	4	4	10,000	Military ± 4%
GREEN	5	5		
BLUE	6	6		
VIOLET	7	7		
GREY	8	8		
WHITE	9	9		
NONE				Military ± 20%
GOLD			0.1 / Mil. Dec. Pt.	Both ± 5%
SILVER			0.01	Both ±10%

6.2 Resistors



Kleur	1e ring	2e ring	3e ring Vermenigv.factor	4e ring Tolerantie
Zwart	0	0	1Ω	
Bruin	1	1	10Ω	± 1%
Rood	2	2	100Ω	± 2%
Oranje	3	3	1kΩ	± 3%
Geel	4	4	10kΩ	± 4%
Groen	5	5	100kΩ	± 0.5%
Blauw	6	6	1MΩ	± 0.25%
Paars/violet	7	7	10MΩ	± 0.10%
Grijs	8	8	100MΩ	± 0.05%
Wit	9	9	1GΩ	
Goud			0.1Ω	± 5%
Zilver			0.01Ω	± 10%

www.resistorguide.com

	Color	Signficant figures			Multiply	Tolerance (%)	Temp. Coeff. (ppm/K)	Fail Rate (%)
Bad	black	0	0	0	x 1		250 (U)	
Beer	brown	1	1	1	x 10	1 (F)	100 (S)	1
Rots	red	2	2	2	x 100	2 (G)	50 (R)	0.1
Our	orange	3	3	3	x 1K		15 (P)	0.01
Young	yellow	4	4	4	x 10K		25 (Q)	0.001
Guts	green	5	5	5	x 100K	0.5 (D)	20 (Z)	
But	blue	6	6	6	x 1M	0.25 (C)	10 (Z)	
Vodka	violet	7	7	7	x 10M	0.1 (B)	5 (M)	
Goes	grey	8	8	8	x 100M	0.05 (A)	1(K)	
Well	white	9	9	9	x 1G			
Get	gold			3th digit only for 5 and 6 bands	x 0.1	5 (J)		
Some	silver				x 0.01	10 (K)		
Now!	none					20 (M)		

