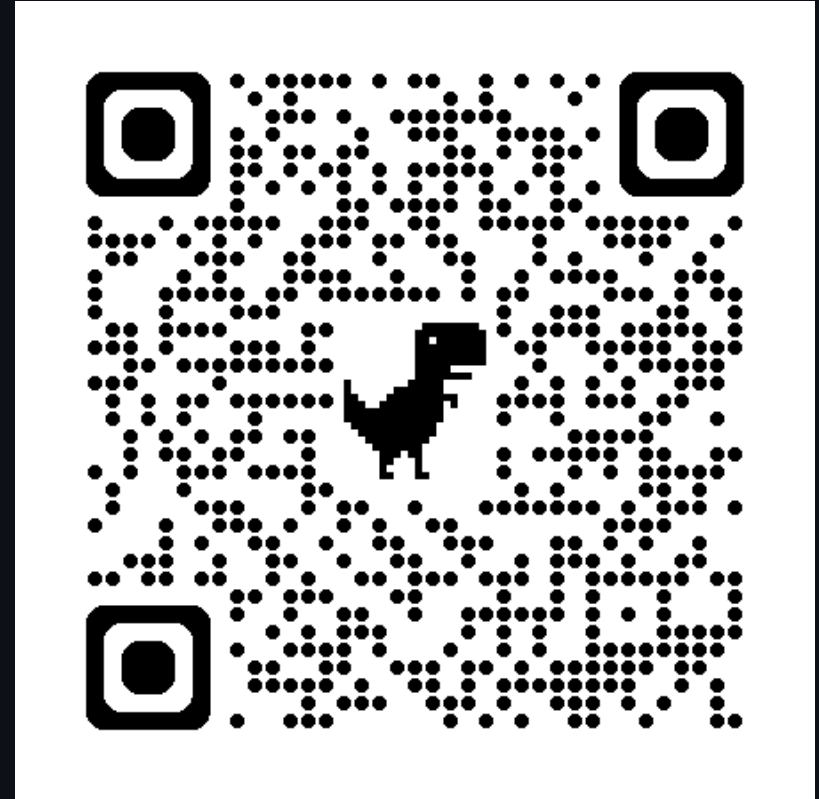


Controlling Servo Motors with the Arduino Uno R4

UB IEEE x DREAM

Please join our Discord:



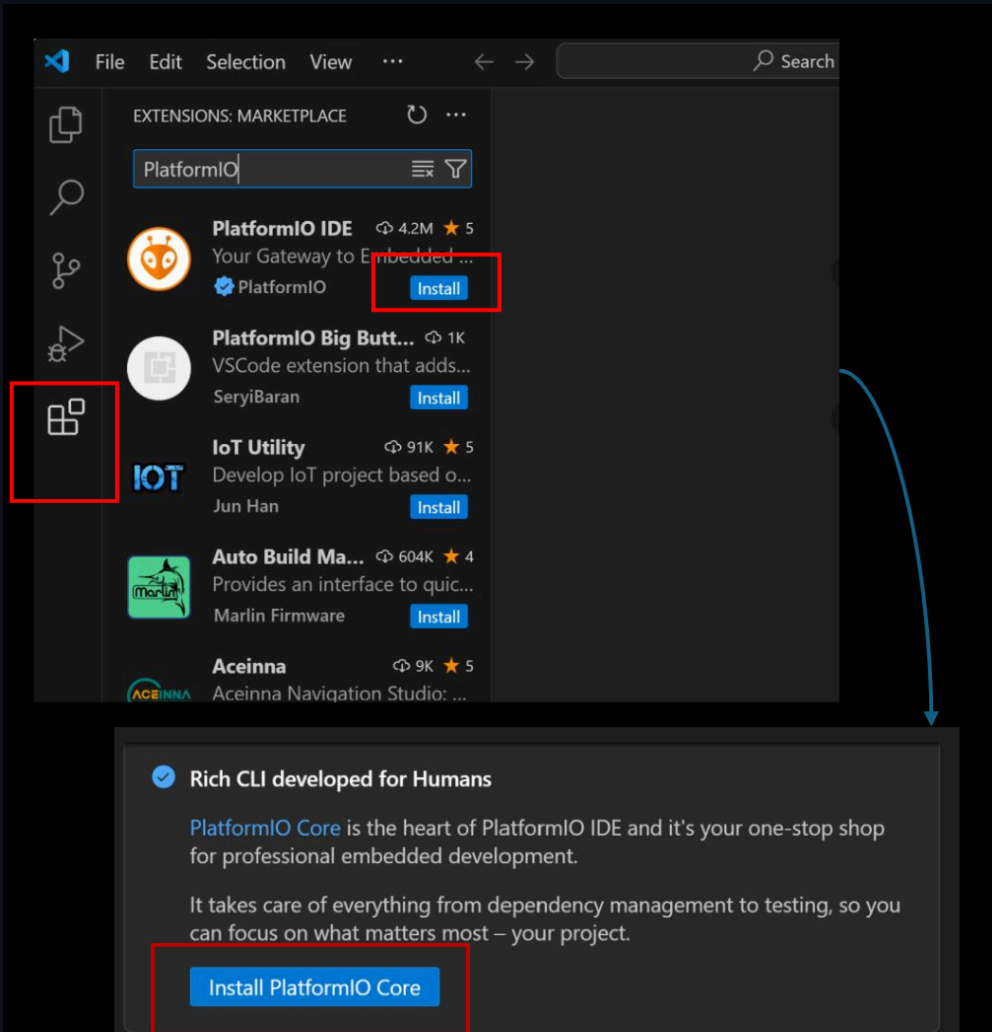
Installing VSCode

Visual Studio Code is a powerful text editor. VS Code's powerful and popular extension ecosystem can give it many of the powers of a fully-fledged IDE. You can download Visual Studio Code from here:
<https://code.visualstudio.com/download>.



Installing PlatformIO

Install PlatformIO as a VSCode extension:



Creating a New Project

The screenshot displays the PlatformIO IDE interface. The top menu bar includes File, Edit, Selection, View, and a search bar labeled 'Search [Administrator]'. The left sidebar contains a 'PROJECT TASKS' section with instructions on opening or creating a project, and a 'QUICK ACCESS' section with links to PIO Home, Open, PIO Account, Inspect, Projects & Configuration, Libraries, Boards, and Platforms. The main workspace shows the 'Welcome to PlatformIO' screen. A red rectangle highlights the '+ New Project' button in the 'Quick Access' section. Other buttons in this section include 'Import Arduino Project', 'Open Project', and 'Project Examples'. The bottom of the screen shows a 'Recent News' section with articles like 'FreeRTOS User Tasks and ISR Code'.

PLATFORMIO

File Edit Selection View ...

Search [Administrator]

PIO Home X

Home

Projects

Inspect

Libraries

Boards

Platforms

PROJECT TASKS

You have not yet opened a PlatformIO project.

You can open an existing PlatformIO-based project (a folder that contains `platformio.ini` file).

Pick a folder

You can create a new PlatformIO Project or explore examples using PlatformIO Home

QUICK ACCESS

PIO Home

Open

PIO Account

Inspect

Projects & Configuration

Libraries

Boards

Platforms

Welcome to PlatformIO

Show at startup

Quick Access

+ New Project

Import Arduino Project

Open Project

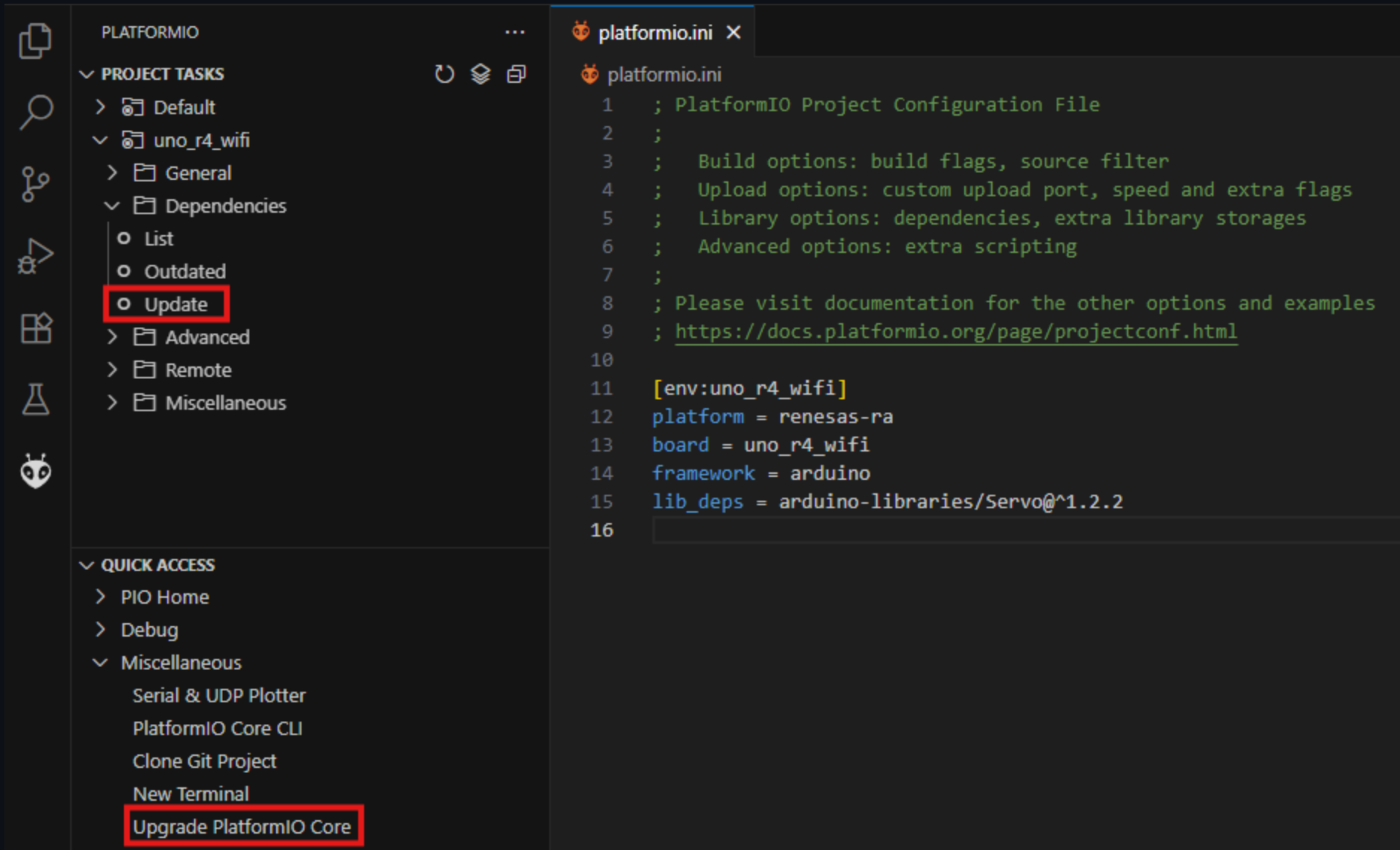
Project Examples

Core 6.1.14 · Home 3.4.4

Recent News

FreeRTOS User Tasks and ISR Code

Upgrade Core + Update Dependencies



The screenshot displays the PlatformIO IDE interface. On the left sidebar, under 'PROJECT TASKS', the 'Dependencies' folder is expanded, and the 'Update' option is highlighted with a red box. Below this, under 'QUICK ACCESS', the 'Upgrade PlatformIO Core' option is also highlighted with a red box. The main editor window shows the 'platformio.ini' file with the following content:

```
platformio.ini
1 ; PlatformIO Project Configuration File
2 ;
3 ; Build options: build flags, source filter
4 ; Upload options: custom upload port, speed and extra flags
5 ; Library options: dependencies, extra library storages
6 ; Advanced options: extra scripting
7 ;
8 ; Please visit documentation for the other options and examples
9 ; https://docs.platformio.org/page/projectconf.html
10
11 [env:uno_r4_wifi]
12 platform = renesas-ra
13 board = uno_r4_wifi
14 framework = arduino
15 lib_deps = arduino-libraries/Servo@^1.2.2
16
```

Arduino Software Convention

Arduino main:

```
int main(void){  
    init();  
    setup();  
    while (true){  
        loop();  
    }  
    return 0;  
}
```

The `platformio.ini` Configuration File

```
[env:uno_r4_wifi]  
platform = renesas-ra  
board = uno_r4_wifi  
framework = arduino
```

Blink - the **Hello, World!** of Hardware

```
#include <Arduino.h>

// put function declarations here:
int myFunction(int, int);

void setup() {
    // put your setup code here, to run once:
    int result = myFunction(2, 3);
}

void loop() {
    // put your main code here, to run repeatedly:
    digitalWrite(LED_BUILTIN, HIGH);
    delay(1000);
    digitalWrite(LED_BUILTIN, LOW);
    delay(1000);
}

// put function definitions here:
int myFunction(int x, int y) {
    return x + y;
}
```


Connecting your Servo Motor

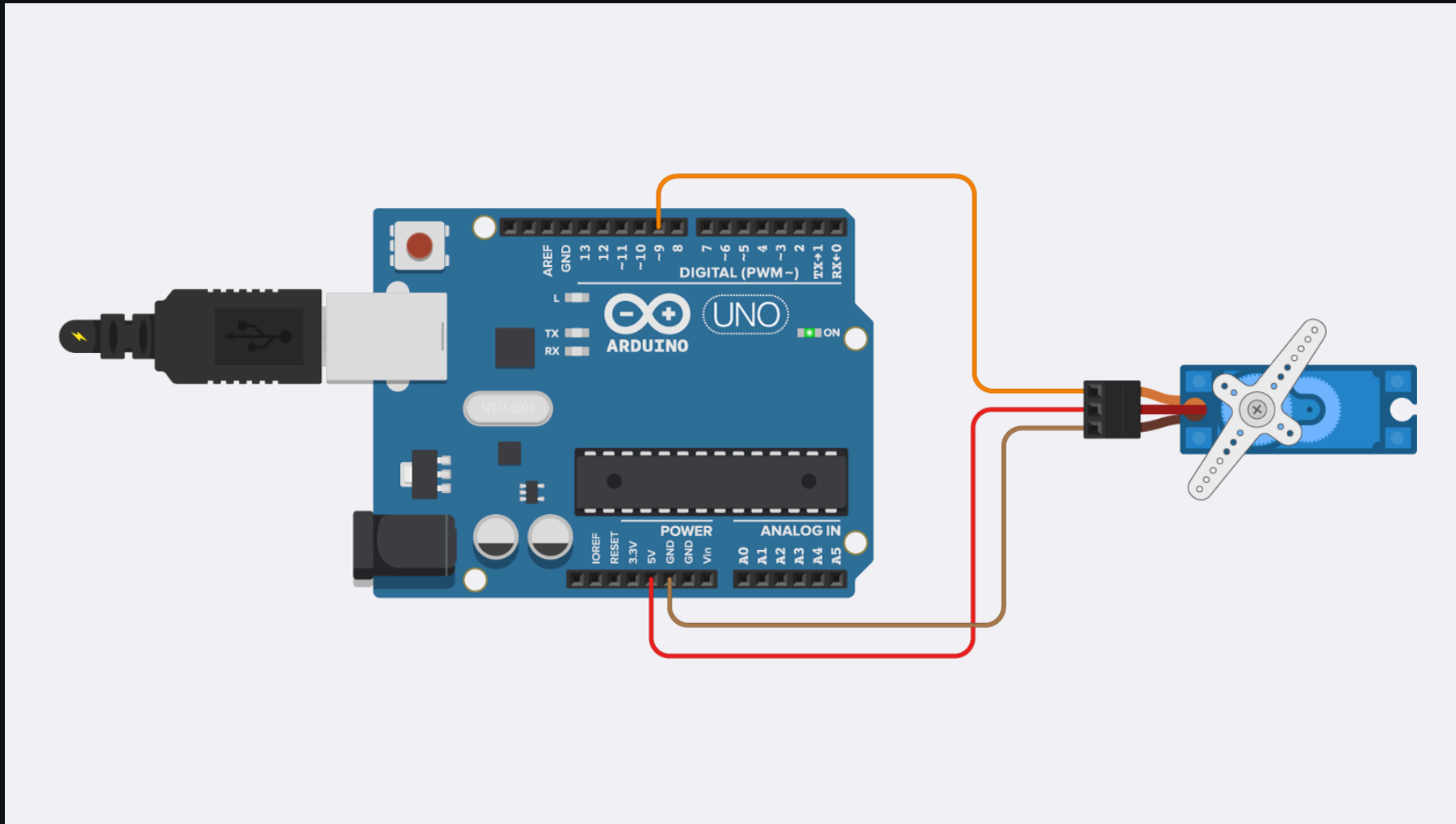


Diagram from [Makerguides](#) by Benne de Bakker, licensed under [CC BY-NC-SA 4.0](#).

Installing Libraries Using PlatformIO

The screenshot displays the PlatformIO IDE interface. On the left, a sidebar contains 'PROJECT TASKS' and 'QUICK ACCESS' menus. The 'Libraries' option in the 'QUICK ACCESS' menu is highlighted with a red box. The main workspace shows the 'Libraries' tab, which includes a search bar with the text 'Servo'. Below the search bar, there are several filter buttons: 'tft display', 'dht*', 'header.RH_ASK.h', 'keyword:mqtt', 'framework:mbed', and 'platform:espressif8266'. The search results list three libraries: 'ESP32Servo' by Kevin Harrington, 'Servo' by Michael Margolis, and 'IRremote' by Armin Joachimsmeier. The 'Servo' library entry is highlighted with a red box. The description for the 'Servo' library states: 'Allows Arduino boards to control a variety of servo motors. This library can control a great number of servos. It makes careful use of timers: the library can control 12 servos using only 1 timer. On the Arduino Due you can control up to 60 servos.' The tags for the 'Servo' library are 'device, control' and 'Atmel AVR, Atmel SAM, Nordic nRF52'.

PLATFORMIO ... PIO Home X platformio.ini

PROJECT TASKS

- > Default
- > uno_r4_wifi
- > General
 - o Build
 - o Upload
 - o Monitor
 - o Upload and Mon...
 - o Clean
 - o Full Clean
 - o Devices
- > Dependencies
- > Advanced
- > Remote
- > Miscellaneous

QUICK ACCESS

- > PIO Home
 - Open
 - PIO Account
 - Inspect
 - Projects & Config...
 - Libraries**
 - Board Libraries
 - Platforms
 - Devices
- > Debug
 - Start Debugging
 - Toggle Debug Co...
- > Miscellaneous
 - Serial & UDP Plotter
 - PlatformIO Core CLI

Home

Projects

Inspect

Libraries

Boards

Platforms

Devices

Registry Installed Built-in Updates

Servo

tft display dht* header.RH_ASK.h keyword:mqtt framework:mbed platform:espressif8266 more...

Libraries 350

ESP32Servo by Kevin Harrington 85,209 6 Arduino

Allows ESP32 boards to control servo, tone and analogWrite motors using Arduino semantics. This library can control a many types of servos. It makes use of the ESP32 PWM timers: the library can control up to 16 servos on individual channels. No attempt has been made to support multiple servos per channel.

device, control

Espressif 32

Servo by Michael Margolis 110,159 2 Arduino

Allows Arduino boards to control a variety of servo motors. This library can control a great number of servos. It makes careful use of timers: the library can control 12 servos using only 1 timer. On the Arduino Due you can control up to 60 servos.

device, control

Atmel AVR, Atmel SAM, Nordic nRF52

IRremote by Armin Joachimsmeier 33,700 28 Arduino

Installing Libraries Using PlatformIO

The screenshot shows the PlatformIO web interface. At the top, there are tabs for 'PIO Home' and 'platformio.ini'. Below the tabs is a navigation bar with icons for Home, Projects, Inspect, Libraries (highlighted), Boards, and Platforms. The main content area displays the 'Servo' library by Michael Margolis. It includes a description, an 'Installation' section with a version dropdown (1.2.2) and an 'Add to Project' button (highlighted with a red box), and a 'More info' link. Below the installation section are tabs for Examples, Installation, Headers, and Changelog. On the right side, there are sections for Tags (device, control) and Platforms (Atmel AVR, Atmel SAM, Nordic nRF52).

PIO Home x platformio.ini

Follow Us

Registry Installed Built-in Updates

Servo by Michael Margolis

Allows Arduino boards to control a variety of servo motors. This library can control a great number of servos. It makes careful use of timers: the library can control 12 servos using only 1 timer. On the Arduino Due you can control up to 60 servos.

Installation

1.2.2 released a year ago

Add to Project | More info

Examples Installation Headers Changelog

Knob

Knob

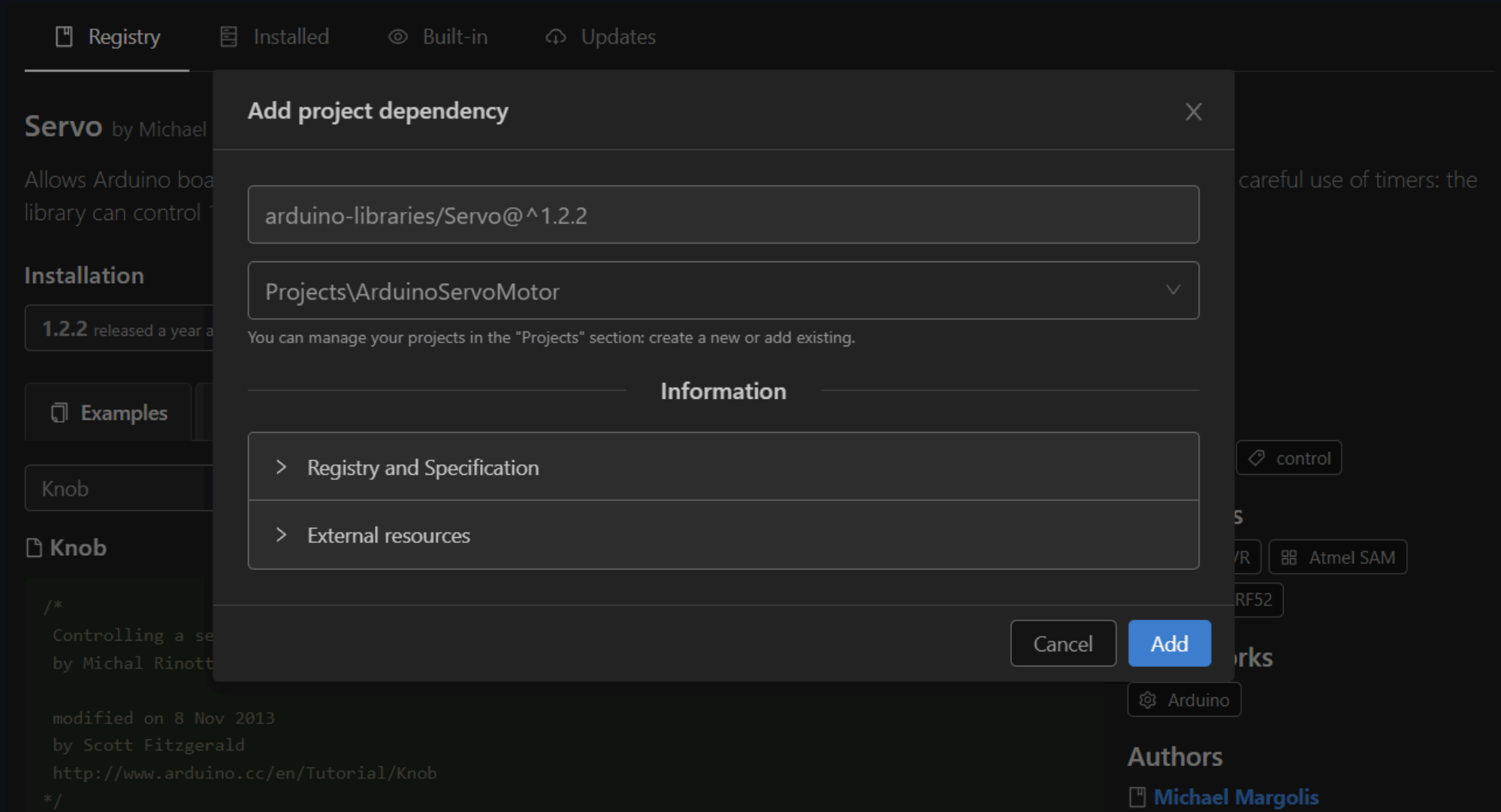
Tags

device control

Platforms

Atmel AVR Atmel SAM Nordic nRF52

Installing Libraries Using PlatformIO



Controlling the Servo

```
#include <Arduino.h>
// Include Servo Library
#include <Servo.h>

Servo myServo;

void setup() {
  // Setup pin 9
  myServo.attach(9);
}

void loop() {
  // control the servo
  myServo.write(0);
  delay(1000);
  myServo.write(90);
  delay(1000);
}
```

Summary of Syntax

Include the Servo Library:

```
#include <Servo.h>
```

Setup your Arduino pin, PWM pins are signaled by "~":

```
myServo.attach(PWM Pin);
```

Control the Servo Motor:

```
myServo.write(Angle Measurement);
```

C++ Looping

```
for (int i = 0; i < 10; i++)  
{  
    // Loop here  
}
```

Coding Challenge

Can you use a loop to gradually change the servo motor?

Remember how to loop from the previous slide, and alter this code to move the servo from 0° to 180°:

```
for (int i = 0; i < 10; i++)  
{  
  // Loop here  
}
```

Once you have the servo rotate from 0° → 180° can you make it go from 180° → 0°?

Hint: You will need to change your starting value, condition, and the increment!

Coding Challenge Answer:

```
for (int pos = 0; pos < 180; pos++)  
{  
    myServo.write(pos);  
    delay(50);  
}  
for (int pos = 180; pos > 0; pos--)  
{  
    myServo.write(pos);  
    delay(50);  
}
```

How do Servo Motors Work?

To answer that question, first we must learn what Pulse Width Modulation (PWM) is!

Pulse Width Modulation (PWM)

PWM is used to control power using digital signals. Instead of varying voltage, electronics use PWM signals. The signal is rapidly switched on and off at a fixed frequency, with a set **duty cycle**. Duty cycle is simply the percentage of time the signal is on.

Duty Cycles

50% duty cycle



75% duty cycle



25% duty cycle



Servo Motor Data Wire

Servo motors operate at 50Hz (20ms period). To control a servo motor, we will send pulses via the signal wire. Every 20ms a pulse is sent with the width of:

- ~0.5ms: 0°
- ~1.5ms: 90°
- ~2.5ms: 180°

writeServo() Function

```
#include <Arduino.h>

const int servoPin = 9;

void writeServo(int angle) {
  // Map angle to pulse width
  int pulseWidth = map(angle, 0, 180, 500, 2400);
  // Signal HIGH for width
  digitalWrite(servoPin, HIGH);
  delayMicroseconds(pulseWidth);
  // Signal LOW for remaining
  digitalWrite(servoPin, LOW);
  delayMicroseconds(20000 - pulseWidth);
}

void setup() {
  pinMode(servoPin, OUTPUT);
}

void loop() {
  for (int i = 0; i < 50; i++) {
    writeServo(0);
  }
  for (int i = 0; i < 50; i++) {
    writeServo(90);
  }
  for (int i = 0; i < 50; i++) {
    writeServo(180);
  }
}
```

Which is better?

It's nice to see how these servo motors work, but this application is unrealistic. There are some errors in our approach:

- Blocking Delays: Can't do anything else because we need to send a constant pulse.
- Pulse Range: Different servo motors have different ranges. We are assuming they are all the same.
- Software Timing: Delays are software-timed, meaning they are not as precise as hardware. This can lead to various jittering issues.

So, we use libraries!

Additional Resources

- VSCode Documentation: <https://code.visualstudio.com/Docs>
- PlatformIO's Documentation: <https://docs.platformio.org/en/latest/>
- Arduino Documentation: <https://docs.arduino.cc/>
- C++ Programming Language: <https://www.learncpp.com/>
- The C Programming Language by Brian Kernighan and Dennis Ritchie: https://en.wikipedia.org/wiki/The_C_Programming_Language
- Purchase the Arduino Uno R4 Wifi (US Store): <https://store-usa.arduino.cc/products/uno-r4-wifi>