Projects based on RPLIDAR A1 360º Laser Range Scanner

Content

Software

Arduino Projects

<ArduinoFolder>\Projects\RPLIDAR_Projects

RPLIDAR_V3_WIFI_DRIVER	05/21/2022	Read RPLIDAR point data from through serial interface and send them via UDP(broadcast.
RPLIDAR_WIFI_BASE_SERVO	05/29/2022	Base station. Receives point data sent by the WiFi driver and keeps a table with the detected points (see RPLIDAT_utils). Also tries to identify points of interest (things that move) and directs a laser pointer towards the point using a servo motor.
CALIBTRATE_SERVO	05/26/2022	Utility to calibrate a servo motor. Allows fine tuning of servo parameters and zero position. The zero degrees direction of the RPLIDAR must align with the 90 degrees direction of the servo.
RPLIDAR_MCP4725_Oscilloscope	03/21/2022	First attempt to interpret RPLIDAR A1 data. It reads directly from the serial port and tries to plot the data on an XY oscilloscope through 2 I2C DACs (MCP4725). It has been tested that and works, but, the result is not very useful. However it serves as an example of the use of RPLIDAR and 2 DACs of this type.
RPLIDAR_WIFI_BASE_STATTION_V2B	05/03/2022	Like RPLIDAR_WIFI_BASE_SERVO but without the servo. The version with servo is derived from this one, in fact.

Project libraries

<ArduinoFolder>\Projects\libraries\RPLIDAR_utils

`RplidarData.h`	05/29/2022	Data structures to hold, update and query the database of points detected by the RPLIDAR. Used by base station programs.
`RPLIDAR_utils.h`	05/03/2022	UDP connection between the driver and the base station(s). The driver communicates by broadcast address. The network is hardcoded in the program. I use a dedicated TP-LINK AP for testing,

RPLIDAR Product Libraries - RPLIDAR Driver Library

Located at:

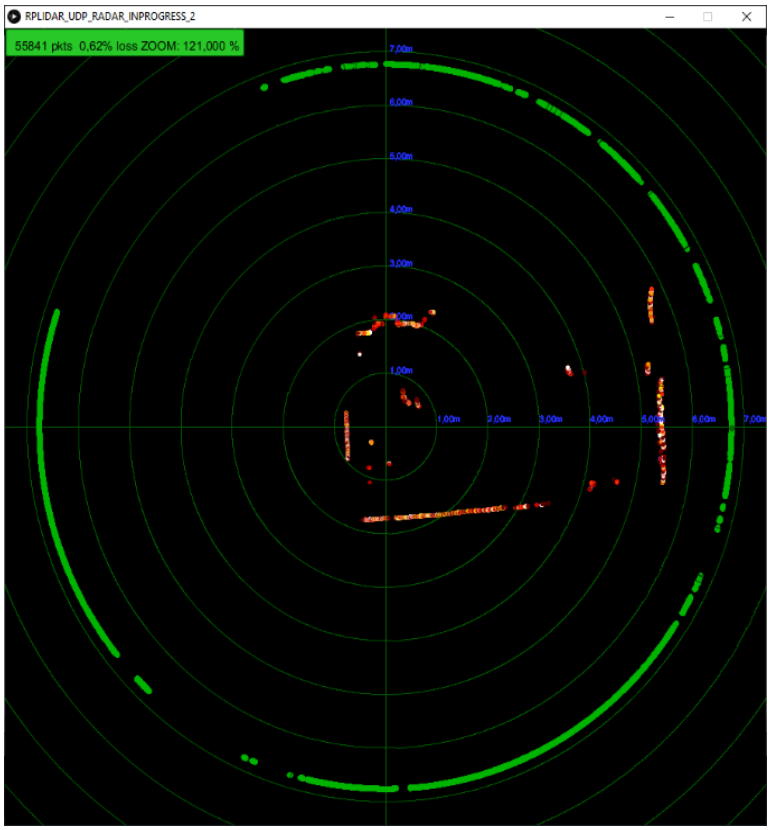
<ArduinoFolder>\Projects\libraries\rplidar_arduino

Source:

https://github.com/robopeak/rplidar_arduino

Processing utilities (V 3.5.4)

\OneDrive - personal\OneDrive\Processing\Projects\RPLIDAR_Projects_Processing



Utilities to visualize the points detected by the RPLIDAR.

WARNING: These programs use a data structure like the one in `RplidarData.h` (the latter is actually derived from these java programs), but they are not necessarily updated with every version, so the algorithms may differ. Be especially careful with the UDP packet structure.

In all of them, use 'h' or '?' for help on options (clicking before on the graphic display).

The colors of the dots indicate the number of times they have been seen (the RPLIDAR does not detect all the dots on every turn).

RPLIDAR_SERIAL_PLOT_POI	05/06/2022	Connect to a base station through a serial port to display the points of interest (POI) detected by the RPLIDAR_data library algorithm. The code must be updated with the COM port assigned to the USB adapter.
RPLIDAR_UDP_ANIMATED	05/22/2022	Visualize data points with animation
RPLIDAR_UDP_RADAR	05/22/2022	Radar display, just for fun
RPLIDAR_UDP_RADAR_INPROGRESS	04/17/2022	Radar display, with modifications
RPLIDAR_UDP_RADAR_INPROGRESS_2	05/22/2022	Display showing moving dots (in blue)
RPLIDAR_UDP_RADAR_INPROGRESS_3	05/22/2022	Similar to v2, also tries (without much success) to display the detected POIs.

Hardware

RPLIDAR A1 (A1M8 MODEL)

https://www.slamtec.com/en/Lidar/A1

There are at least three versions of the device, one (rev 1.0) with a datasheet dated 2016, another (rev 3.0) from 2020. I have not been able to access version 2.0 datasheet. Different versions have slightli different connections. The project is done with the 2020 one, but it should work with others.

2020 Version:

> https://bucket-download.slamtec.com/d1e428e7efbdcd65a8ea111061794fb8d4ccd3a0/LD108_SLAMTEC_rplidar_datasheet_A1M8_v3.0_en.pdf

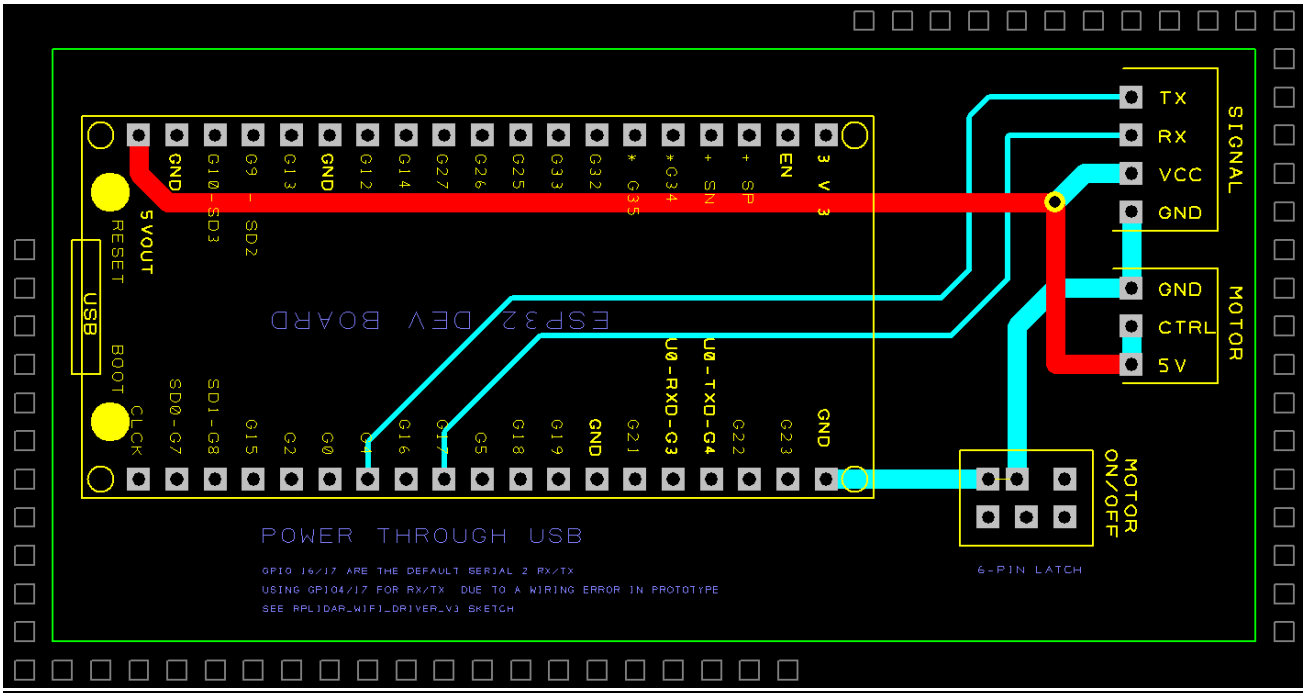
2016 Version:

> https://bucket-download.slamtec.com/e680b4e2d99c4349c019553820904f28c7e6ec32/LM108_SLAMTEC_rplidarkit_usermaunal_A1M8_v1.0_en.pdf

The RPLIDAR accepts a motor speed PWM control signal. I have set it to 5V DC fixed, maximum speed. Important, communication with RPLIDAR does not work if the 5V input to the device's serial interface is not provided by the microcontroller but instead is taken from an external power supply.

Driver

The driver circuit is very simple. It is a breakout board with the pins that the RPLIDAR needs, in the same order.



Base station with servo

The circuit used in the project uses an ESP01 microcontroller (Generic ESP8266 Arduino core). The circuit assumes a 5V input and provides 3.2 V for the ESP01 through an LM1117 regulator. An NodeMCU or ESP32 boards should work as well. It also provides a socket for an USB to Serial adapter. **Warning**: use a 3.3 V capable adaptor. The wiring might differ depending on the adaptor used.

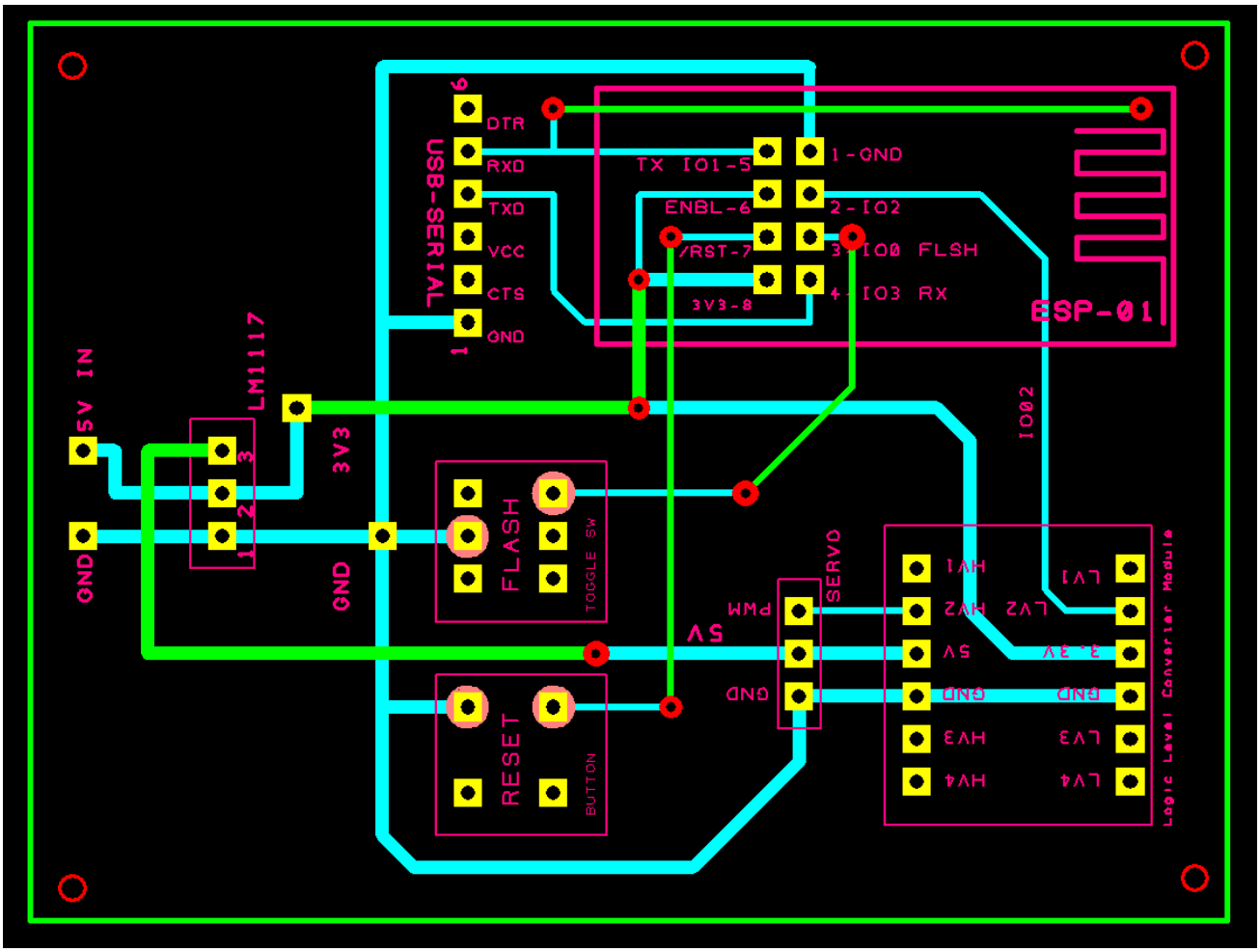
Some models of servo motor might need tweaking the parameters that control position. These are defined in the program as MIN_SERVO_US and MAX_SERVO_US. See the servo library attach() method:

```
Servo.attach(pin, min, max)
```

`min`: the pulse width, in microseconds, corresponding to the minimum (0 degree) angle on the servo (defaults to 544)

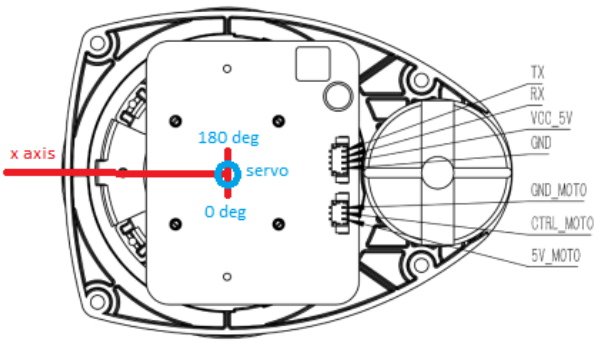
`max`: the pulse width, in microseconds, corresponding to the maximum (180 degree) angle on the servo (defaults to 2400)

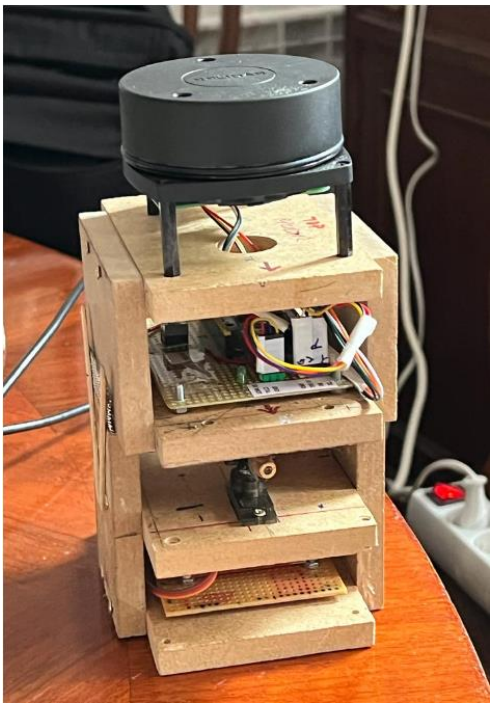
Use the **`CALIBRATE_SERVO.ino`** sketch to find the correct values for your servo.



Notes on hardware setup

The software assumes that the servo is mounted underneath the RPLIDAR, with the servo's axis aligned with the RPLIDAR's spindle. The servo's 90% position must be aligned with the RPLIDAR's x-axis (line passing through the rotation axis and the point in the middle of the space between the longest legs).





This disposition may cause interference between the WiFi modules of the base station and the driver. The interference might be reduced by placing a piece of wire mesh between the devices.

And improved design might consider the relative position of the RPLIDAR's spindle and the servo's axis (3 numbers: distance, azimuth, and relative rotation of the axes) and then resolve the resulted triangle for the adjusted servo's orientation.

Software usage notes

The include file `RPLIDAR_utils.h` contains the WiFi parameters and the `RPLIDAR_Packet` message structure used in UDP communications between the driver and the base station.

It also includes logging functions with several log levels and some wrappers for `Serial.print()` and `Serial.println()` to facilitate redirecting or disabling console output.

`RplidarData.h` is used only by the base station. The user must declare a `LidarData` object and then call the `set()` method for every packet received form the driver.

The `loop()` method **must** be called frequently, preferably with each invocation of the sketch's `loop()` function. It performs the `LidarData` structure housekeeping tasks

Then the `get()` and `getPointOfInterest()` methods can be used to retrieve the distance information for a given angle and the data of the most recently moved point. Both functions use a `DistanceData` object (also declared in `RplidarData.h`) to hold the returned data.

The header file also contains parameters configuring angular resolution, persistence of detected points and motion detection criteria. Too much angular resolution will make the `LidarData` structure too big to fit in memory.

By default, points further than `DEFAULT_MAX_POI_DIST` (currently 5 meters) are not candidates to be considered as points of interest. This may be changed with the `setMaxPOIDist()` method.

`void loop()`

Perform `LidarData` structure housekeepking. Call whenever the calling program is idle.

`void setMaxPOIDist(float max)`

Change max distance at which POIs are looked for.

`bool set(float angle, float distance, int quality, long sequence)`

Set the position data for a given angle. Returns true if the point has moved.

`bool get(float angle, DistanceData& d, bool erase)`

Get position data for a given angle. Returns true if there is valid data recorded. Optionally, erase (invalidate) the data.

`bool getPointOfInterest(DistanceData &dd)`

Get point data for the point of interest (most recently moved point with certain conditions). Returns true if there is a point of interest, false otherwise.

`float getLoss()`

Get percentage of UDP packets lost.

`long getPackets()`

Get number of UDP packets received.