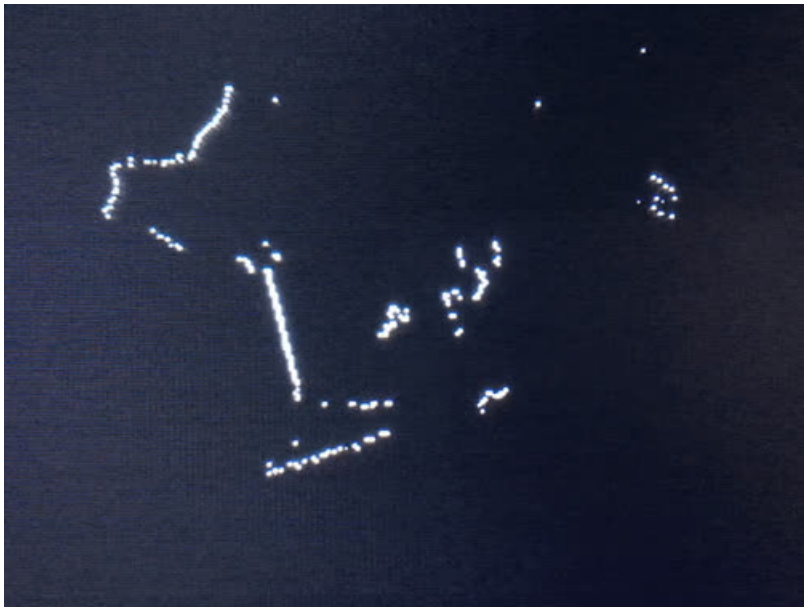
adafruit learning system

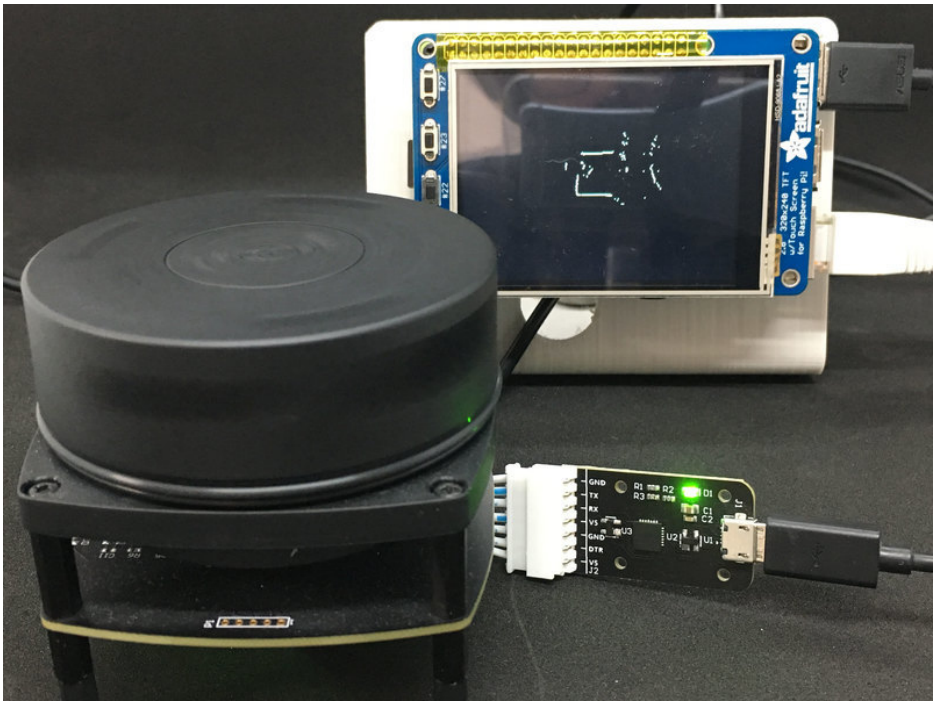
Using the Slamtec RPLIDAR on a Raspberry Pi

Created by Dave Astels



Last updated on 2020-08-07 03:03:26 PM EDT

Overview



LIDAR is all the range*. It's one of the fundamental sensing technologies of autonomous vehicles. It gives a system the ability to see how far away things are 360 degrees around it.

We used the Garmin LIDAR range finder in another guide (https://adafru.it/E7L). It works by sending out a pulse of modulated laser light which bounces off whatever it's aimed at and is read by an optical receiver. The time that takes is a measure of distance to the surface it bounced off of. While being very cool and very effective technology, it just looks in one direction. You could mount it on a stepper motor and repeatedly rotate and take a reading. Or you could use a 360 degree LIDAR system like the RPLIDAR from Slamtec.

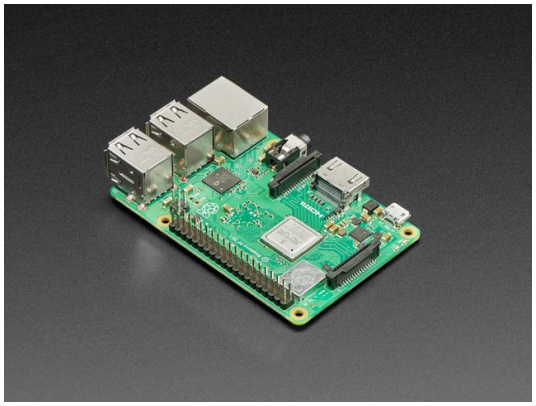
* See what I did there?

Parts

Slamtec RPLIDAR A1 - 360 Laser Range Scanner

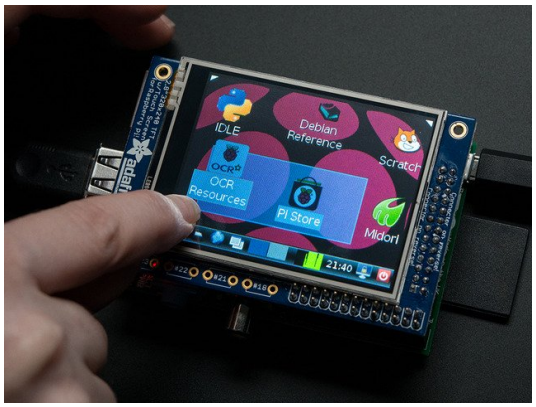
$114.95
IN STOCK

Add To Cart



Raspberry Pi 3 - Model B+ - 1.4GHz Cortex-A53 with 1GB RAM

$35.00
IN STOCK
Add To Cart



Adafruit PiTFT - 320x240 2.8" TFT+Touchscreen for Raspberry Pi

$34.95
IN STOCK
Add To Cart

USB cable - USB A to Micro-B

$2.95
IN STOCK
Add To Cart

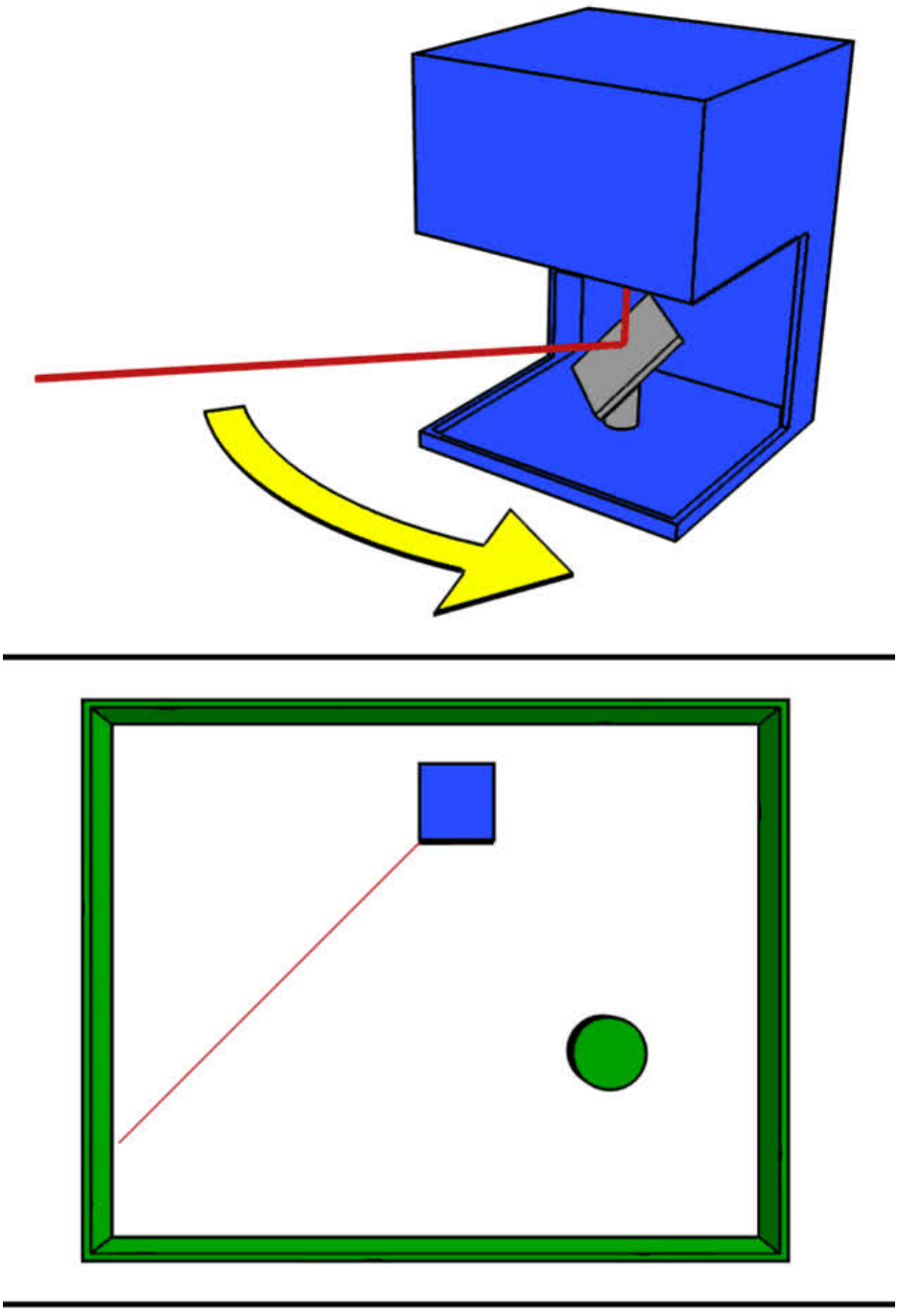
5V 2.5A Switching Power Supply with 20AWG MicroUSB Cable

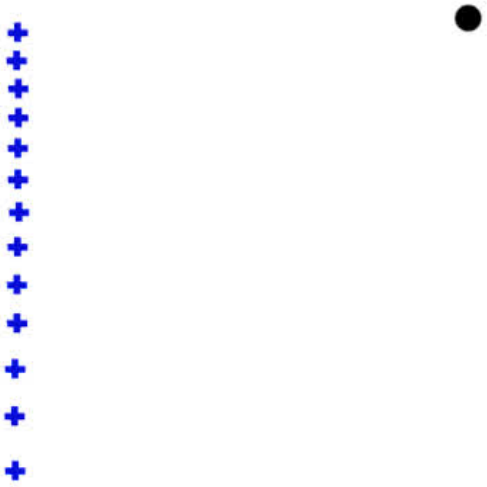
$7.50
IN STOCK
Add To Cart

360 Degree LIDAR

Instead of taking a step-and-read approach, the RPLIDAR drives the rotating scanner with a DC motor, continuously taking readings and making note of the scanner's angle with each one. Because of this approach, a single revolution is not guaranteed to give a reading for each possible angle, but over several rotations a full scan can be assembled.

The animation below shows a different mechanical approach (using a rotating mirror instead of a rotating sensor) and only covers 180 degrees, but the concept is identical. It shows the general theory of operation: readings taken at different angles, combined to provide a map of distances from the sensor. Using this data a system can get a sense of the space around it.





Slamtec RPLIDAR A1 - 360 Laser Range Scanner

$114.95
IN STOCK

Add To Cart

The device interfaces through a serial connection and comes with a USB to serial adaptor, not unlike an FTDI adapter **(but not the same pinout as one).**

The bundled USB adaptor is functionally similar to an FTDI adapter, but does not have the same connections.

As mentioned, the RPLIDAR interfaces using serial. Rx and Tx for data, and a digital input to turn the motor on and off. You can use these directly, or with the included USB to serial adapter.

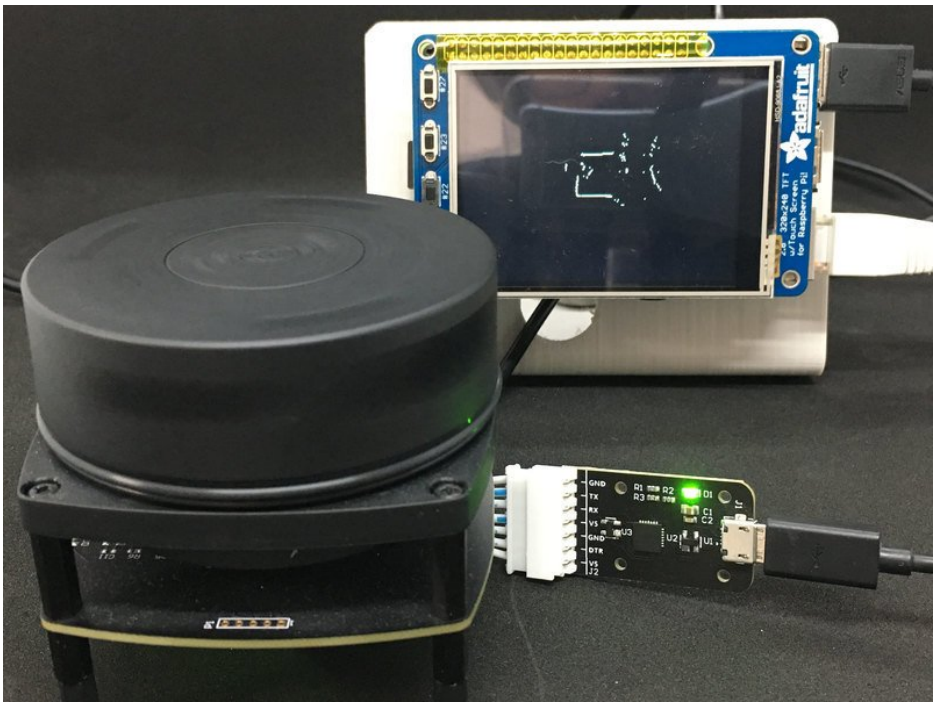
Get started by installing the module on the Raspberry Pi using pip:

```
pip install adafruit-circuitpython-rplidar
```

For now this supports CPython (well tested on the Raspberry Pi). Support for CircuitPython on M4 boards will be

forthcoming.

CPython on Raspberry Pi



The simplest way to use the RPLIDAR with a Raspberry Pi is to interface through the bundled USB adapter. Connect a USB cable on the Raspberry Pi and then plug the other end into the LIDAR USB adapter board. Ensure the adapter board is fitted correctly onto the sensor per the instructions in the manual.

PyGame

We'll be using the 2.8" PiTFT display. See this guide (https://adafru.it/dDK) for instructions on installing and configuring the required support software. We'll also need pyGame to provide that data plotting capabilities. See the pyGame site for installation instructions (https://adafru.it/E7M).

The Code

To use the RPLIDAR, you need to start the motor spinning and tell it to start taking readings. It will then stream each reading it takes until you tell it to stop. Each reading includes the distance sensed, and the angle of the reading. To make things easier, the Skoltech library provides a convenient wrapper for this functionality, and that's what we'll use for this guide.

This example will consume data from the RPLIDAR and display it on a 2.8" PiTFT display. We start by setting up a few things.

```
import os
from math import cos, sin, pi, floor
import pygame
from adafruit_rplidar import RPLidar

# Set up pygame and the display
os.putenv('SDL_FBDEV', '/dev/fb1')
pygame.init()
lcd = pygame.display.set_mode((320,240))
pygame.mouse.set_visible(False)
lcd.fill((0,0,0))
pygame.display.update()

# Setup the RPLidar
PORT_NAME = '/dev/ttyUSB0'
lidar = RPLidar(None, PORT_NAME)

# used to scale data to fit on the screen
max_distance = 0
```

Next we create a buffer in which to keep distance data. Each item in the list stores the distance measured at each degree for a complete rotation. We use this to maintain the most recent set of measurements even though each scan will be incomplete.

```
scan_data = [0]*360
```

We use the `iter_scans` function of the RPLIDAR object. This starts the motor and the scanning for us so we can skip that in our code. `iter_scans` accumulates measurements for a single rotation and returns a list of tuples, from which we are interested in the second and third items. These are the angle and the distance measured at that angle. Both are floating point values.

Once we have a scan, we step through each data point. The floor of the angle is taken and used as the index at which to store the measurement value. Because it is possible that multiple consecutive measurements in a scan could be close to the same angle (and thus having the same floor value) there is a small chance that some measurements are lost. But there is no real disadvantage to this, so it can be disregarded.

```
scan_data = [0]*360

try:
    print(lidar.info)
    for scan in lidar.iter_scans():
        for (_, angle, distance) in scan:
            scan_data[min([359, floor(angle)])] = distance
        process_data(scan_data)

except KeyboardInterrupt:
    print('Stoping.')
lidar.stop()
lidar.disconnect()
```

Once we have updated `scan_data`, it is passed to the `process_data` function.

The `process_data` function can do anything from finding the closest object, to choosing the best direction to move in.

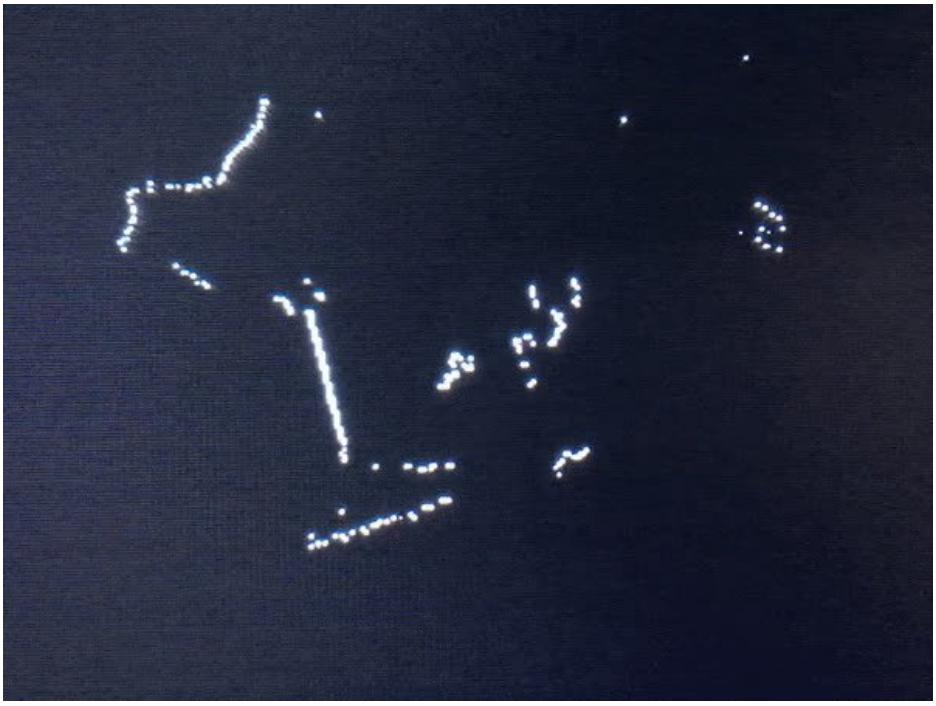
The only requirement is that it be as fast as possible. If it takes too long to process a scan, data from the RPLIDAR will eventually be dropped. Since we're constantly reading fresh data, dropping an occasional scan shouldn't be an issue.

In this example we display the distance data on a PiTFT.

The first step of a pass is to clear the display. Then it looks at each angle: 0-359. If the distance recorded for that angle is zero, it's because a reading hasn't be acquired for that angle yet, and we can safely ignore it. If there is a distance recorded for the angle, `max_distance` is adjusted as necessary to keep the display scaled to fit on the screen. The angle is converted to radians (from degrees) and the vector for the reading is converted to a cartesian coordinate which is then used to plot the data point on the display.

After all measurements have been plotted, the display is updated.

```
def process_data(data):
    global max_distance
    lcd.fill((0,0,0))
    for angle in range(360):
        distance = data[angle]
        if distance > 0:                  # ignore initially ungathered data points
            max_distance = max([min([5000, distance]), max_distance])
            radians = angle * pi / 180.0
            x = distance * cos(radians)
            y = distance * sin(radians)
            point = (160 + int(x / max_distance * 119), 120 + int(y / max_distance * 119))
            lcd.set_at(point, pygame.Color(255, 255, 255))
    pygame.display.update()
```



Full code

```
"""
Consume LIDAR measurement file and create an image for display.
```

```
Adafruit invests time and resources providing this open source code.
Please support Adafruit and open source hardware by purchasing
products from Adafruit!

Written by Dave Astels for Adafruit Industries
Copyright (c) 2019 Adafruit Industries
Licensed under the MIT license.

All text above must be included in any redistribution.
"""

import os
from math import cos, sin, pi, floor
import pygame
from adafruit_rplidar import RPLidar

# Set up pygame and the display
os.putenv('SDL_FBDEV', '/dev/fb1')
pygame.init()
lcd = pygame.display.set_mode((320,240))
pygame.mouse.set_visible(False)
lcd.fill((0,0,0))
pygame.display.update()

# Setup the RPLidar
PORT_NAME = '/dev/ttyUSB0'
lidar = RPLidar(None, PORT_NAME)

# used to scale data to fit on the screen
max_distance = 0

#pylint: disable=redefined-outer-name,global-statement
def process_data(data):
    global max_distance
    lcd.fill((0,0,0))
    for angle in range(360):
        distance = data[angle]
        if distance > 0:                  # ignore initially ungathered data points
            max_distance = max([min([5000, distance]), max_distance])
            radians = angle * pi / 180.0
            x = distance * cos(radians)
            y = distance * sin(radians)
            point = (160 + int(x / max_distance * 119), 120 + int(y / max_distance * 119))
            lcd.set_at(point, pygame.Color(255, 255, 255))
    pygame.display.update()


scan_data = [0]*360

try:
    print(lidar.info)
    for scan in lidar.iter_scans():
        for (_, angle, distance) in scan:
            scan_data[min([359, floor(angle)])] = distance
        process_data(scan_data)

except KeyboardInterrupt:
    print('Stoping.')
lidar.stop()
```

```
lidar.disconnect()
```

The Road Ahead

We have a Raspberry Pi consuming data from the RPLIDAR and displaying it on a screen, using the Skoltech library.

There are several possible ways forward. Continuing on the Pi, different applications can be explored using the LIDAR data. Robot navigation would be a fun one.

In a different direction, the library can be ported to CircuitPython and run on SAMD boards. Look forward to seeing that added to this guide in the near future.