How To Pass - Foundation Apprenticeship Year 2

Dynamic Home Computational Probe

Whitepaper

Dynamic Home Computational Probe (DHCP)

Design Brief

Our design is to create an autonomous robot that will – using OpenCV and ROS, navigate its environment effectively and to provide a sensor platform for data and readings in remote locations, due to its effective and simple rollout of Amateur Digital Radio for VHF communications. This effectively means that it will increase range over traditional Wi-Fi / Bluetooth IOT devices thousand-fold. Data integrity is also upheld through effective 2-Bit Error correction to ensure all mission-critical data is received and transmitted correctly. Its reliability and range makes it an effective devices from any scale – such as an office or even in outdoor expeditions where human presence could be endangered.

Packet Radio

- Uses our own cursed data protocols - **DNS** (**D**ata **N**etwork **S**olution)
- Runs over Direwolf
- Station Transciever - Baofeng UV-5R (1 W)
- Probe Transciever - Yaesu Vertex VX-150 (100mw)

Getting Started

Dependencies

All dependencies are in requirements.txt

```
rpi-gpio==0.6.5 X
Bluetin-Echo==0.2.0 X
python3-smbus X
```

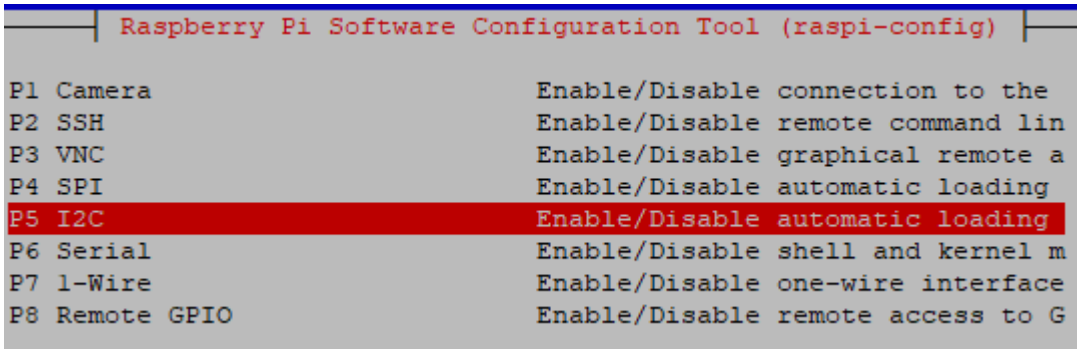
Installation

The Raspberry Pi is going to need some initial configuration to run the code. Also, to support the SMBus2 Python library, we need to install some dependencies.

Firstly, we configure the Rasberry Pi to enable the I2C interface. So, open a Terminal session on the Raspberry Pi to execute the following command:

```
sudo raspi-config
```

The Raspberry Pi Configuration Tool will now open to allow you to select interface options. Then on the next screen, choose I2C to enable the interface.



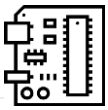
Secondly, we now install the dependencies required by the SMBus2 library. execute the following in the Pi Terminal:

```
sudo apt-get install i2c-tools
```

Lastly, we install SMBus2, which is a drop-in replacement for smbus-cffi/smbus-python in pure Python. Enter the following command in the Terminal:

```
# Either for Python 2.7
pip install smbus2
# Or, for Python 3
pip3 install smbus2
```

The Circuit

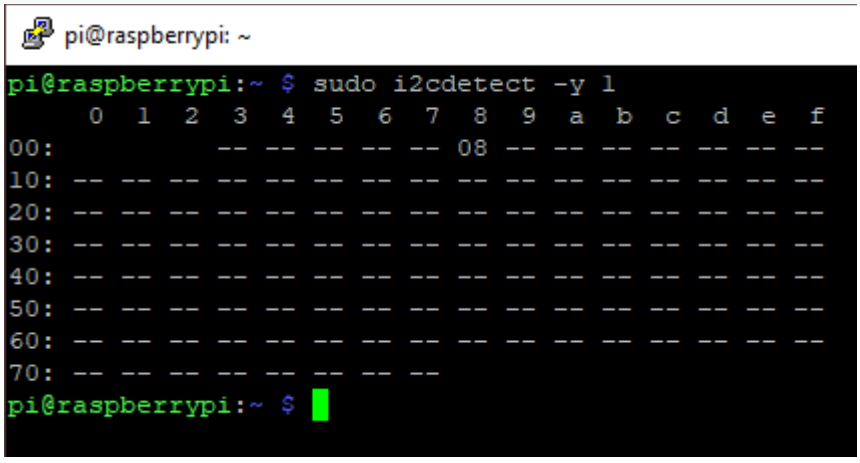


To run the example code, you will need to connect the I2C interface between the Raspberry Pi and Arduino. The Raspberry Pi will have the USB power, while the Arduino has power from the computer USB serial connection. The Arduino is a 3.3v Pro Mini compatible running at 8MHz. And due to the separate power supplies, we need to connect the grounds between the Arduino and Raspberry Pi.

However, this circuit will not be compatible with 5v Arduino board versions. While it is possible to disable the Arduino pull-up resistors, a 5v spike on the I2C is still possible during the pull-up transition.

So, to check the correct installation of the software packages, run the following command at the Terminal Prompt. However, you will need the code uploaded and running on the Arduino. This example code will set the slave address of the Arduino board, and hopefully, this test will detect that address.

```
sudo i2cdetect -y 1
```



ROS Setup

Configure your Ubuntu repositories

Configure your Ubuntu repositories to allow "restricted," "universe," and "multiverse." You can follow the Ubuntu guide for instructions on doing this.

Setup your sources.list

Setup your computer to accept software from packages.ros.org.

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main"
> /etc/apt/sources.list.d/ros-latest.list'
```

Set up your keys

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key
C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

If you experience issues connecting to the keyserver, you can try substituting hkp://pgp.mit.edu:80 or hkp://keyserver.ubuntu.com:80 in the previous command.

Alternatively, you can use curl instead of the apt-key command, which can be helpful if you are behind a proxy server:

```
curl -sSL
'http://keyserver.ubuntu.com/pks/lookup?op=get&search=0xC1CF6E31E6BADE8868B172B4F4
2ED6FBAB17C654' | sudo apt-key add -
```

Installation

First, make sure your Debian package index is up-to-date:

```
sudo apt update
```

There are many different libraries and tools in ROS. We provided four default configurations to get you started. You can also install ROS packages individually.

In case of problems with the next step, you can use following repositories instead of the ones mentioned above ros-shadow-fixed

Desktop-Full Install: (Recommended) : ROS, rqt, rviz, robot-generic libraries, 2D/3D simulators and 2D/3D perception

`sudo apt install ros-melodic-desktop-full`

Desktop Install: ROS, rqt, rviz, and robot-generic libraries

`sudo apt install ros-melodic-desktop`

ROS-Base: (Bare Bones) ROS package, build, and communication libraries. No GUI tools.

```
sudo apt install ros-melodic-ros-base
```

Individual Package: You can also install a specific ROS package (replace underscores with dashes of the package name):

```
sudo apt install ros-melodic-PACKAGE
```

e.g.

```
sudo apt install ros-melodic-slam-gmapping
```

To find available packages, use:

```
apt search ros-melodic
```

Initialize rosdep

Before you can use ROS, you will need to initialize rosdep. rosdep enables you to easily install system dependencies for source you want to compile and is required to run some core components in ROS.

```
sudo rosdep init
rosdep update
```

Environment setup

It's convenient if the ROS environment variables are automatically added to your bash session every time a new shell is launched:

```
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

If you have more than one ROS distribution installed, ~/.bashrc must only source the setup.bash for the version you are currently using.

If you just want to change the environment of your current shell, instead of the above you can type:

```
source /opt/ros/melodic/setup.bash
```

If you use zsh instead of bash you need to run the following commands to set up your shell:

```
echo "source /opt/ros/melodic/setup.zsh" >> ~/.zshrc
source ~/.zshrc
```

Dependencies for building packages

Up to now you have installed what you need to run the core ROS packages. To create and manage your own ROS workspaces, there are various tools and requirements that are distributed separately. For example, rosinstall is a frequently used command-line tool that enables you to easily download many source trees for ROS packages with one command.

To install this tool and other dependencies for building ROS packages, run:

```
sudo apt install python-rosinstall python-rosinstall-generator python-wstool
build-essential
```

Built With

- Direwolf - The Packet radio system implemented

Versioning

We use SemVer for versioning. For the versions available, see the tags on this repository.

Authors

- **Logan Tarvit** - *Packet radio system implementation, Sensor platform creation and data transportation across the device, General hardware construction, Webserver Implementation.* - Axiom
- **Rory Cormack** - *AI construction and maintenance, SQL Database construction, High and low level movement processing* - Liather

License

This project is licensed under the GPL 3.0 License - see the LICENSE.md file for details

Acknowledgments

- BlueTin.IO
- W3Schools
- Fritzing
- Dire Wolf
- I2C Arduino to Pi
- Communication Standards PDF
- I2C Logo (Philips (NXP))
- Circuit SVG