ECE 492 Capstone Project

SmartAVL Raspberry Pi Setup

**Setting up Raspbian**

Raspbian is the recommended operating system for the Raspberry Pi, and as such, Raspbian is the operating system that is supported by our SmartAVL project. It can be installed as follows:

1. Connect a microSD card with a storage capacity of 16 gigabytes or more to your computer.
2. Install [NOOBS](https://www.raspberrypi.org/downloads/noobs/) on the microSD card.
3. Plug the microSD card into the Raspberry Pi, then connect a mouse, keyboard, and display to the Raspberry Pi.
4. Power up the Raspberry Pi and follow the onscreen prompts to install Raspbian.

**Setting up PiCAN**

The PiCAN board is an attachment to the Raspberry Pi. It can be installed by following [this guide](https://copperhilltech.com/pican2-controller-area-network-can-interface-for-raspberry-pi/). When soldering the board, ensure that the termination resistor is activated, and that the DB-9 connector is configured for an OBD-II cable.

**Setting up Python-CAN**

The Smart AVL device uses Python, and relies on the Python-CAN library, which is not installed by default. It can be installed and configured as follows:

1. Open a new terminal window on the Raspberry Pi.
2. Execute the command **pip3 install python-can**
3. Execute the command **nano ~/can.conf**
   1. This command opens an editor window. Paste the following into the editor window, then save your changes:

[default]

interface = socketcan

channel = can0

bitrate = 500000

**Setting up Bluetooth**

There are a number of steps to properly setup Bluetooth capabilities for the Raspberry Pi. This guide will explain how to setup the official Raspberry Pi Bluetooth library for python. We also strongly recommend setting up the GUI for managing Bluetooth devices and pairing with the Raspberry Pi. All of these instructions should be run on the Raspberry Pi.

1. Open up a new terminal window.
2. Execute **sudo apt-get install bluetooth bluez blueman pi-bluetooth**. This will install the GUI as well as the underlying Bluetooth libraries that we will be using.
3. Reboot the Raspberry Pi.
4. Ensure Bluetooth has been started with **systemctl status Bluetooth**. If Bluetooth is not active, turn it on with **sudo systemctl start Bluetooth**.
5. The next following steps 6-12 detail how to resolve a permission error if an “errno 13: Permission Error” is encountered during operation. We recommend this step is done now as a precaution. Credit to “dlech” for solving this error (<https://github.com/ev3dev/ev3dev/issues/274>)
6. In **/etc/systemd/system/bluetooth.service** edit the “ExecStart” line by appending the flag **“--compat**” on that line.
7. Add the “pi” user to the Bluetooth group with **sudo usermod +aG Bluetooth pi**. If this fails, add the Bluetooth group with **newgrp Bluetooth** and try the above command again.
8. The next three steps are for having these changes persist on boot.
9. Create a new file **/etc/systemd/system/var-run-sdp.path** and add the following text:

[Unit]

Descrption=Monitor /var/run/sdp

[Install]

WantedBy=bluetooth.servicae

[Path]

PathExists=/var/run/sdp

Unit=var-run-sdp.service

1. Create a new file **/etc/systemd/system/var-run-sdp.service** and add the following text:

[Unit]

Description=Set permission of /var/run/sdp

[Install]

RequiredBy=var-run-sdp.path

[Service]

Type=simple

ExecStart=/bin/chgrp bluetooth /var/run/sdp

1. Start the new daemon with the following commands:
   1. Run **sudo systemctl daemon-reload**.
   2. Run **sudo systemctl enable var-run-sdp.path**.
   3. Run **sudo systemctl enable var-run-sdp.service**.
   4. Run **sudo systemctl start var-run-sdp.path**.
2. Reboot the Raspberry Pi and Bluetooth will now be setup.

**Setting up the Adafruit Breakout GPS**

To take advantage of Adafruit’s libraries that they package with their hardware perihperals, multiple steps will have to be taken to configure the Raspberry Pi. This involves setting up I2C, SPI and the GPIO pins and installing Adafruit’s CircuitPython library. This guide is taken from <https://learn.adafruit.com/circuitpython-on-raspberrypi-linux/installing-circuitpython-on-raspberry-pi>

1. Enable I2C support on the Raspberry Pi.
   1. Install the I2C-Tools utility with **sudo apt-get install -y python-smbus** and **sudo apt-get install -y i2c-tools**.
   2. Configure the Raspbian Kernel by opening up the configuration menu with **sudo raspi-config**.
   3. In the Configuration Tool, navigate to Interfacing Options > I2C. Select yes to enable the “ARM I2C interface” and yes for the “I2C kernel to be loaded by default”.
   4. Reboot the Raspberry Pi. Test by running the command **sudo 12cdetect –y 1** which should show an address map of active I2C addresses in use. On a new installation it is normal for this to be completely blank.
2. Enable SPI support on the Raspberry Pi.
   1. Configure the Raspbian Kernel by opening up the configuration menu with **sudo raspi-config**.
   2. In the Configuration Tool, navigate to Interfacing Option > SPI. Select yes to enable the “SPI interface to be enabled”.
   3. Reboot the Raspberry Pi. Test by running the command **ls -l /dev/spidev\***, there should be one or two devices shown.
3. Now it is time to configure the device based on how the GPS is being connected to the Raspberry Pi
   1. If connecting directly using GPIO pins follow the following instructions.
      1. Configure the Raspbian Kernel by opening up the configuration menu with **sudo raspi-config**.
      2. In the Configuration Tool, navigate to Interfacing Option > Serial. Select no to “enabling a login shell over serial” and yes to having “the serial port hardware to be enabled”.
      3. Reboot the Raspberry Pi. Ensure the serial monitor has been enabled by confirming that the device at path **/dev/ttyS0** is present.
   2. Else if connecting via USB use the following instructions.
      1. Add the pi user to the “dialout” group by **sudo usermod –a –G dialout pi**.
      2. Note this assumes that the GPS is the only USB device plugged into the Pi. Otherwise the gpsClass.py file may need to be changed to connect to the correct USB port.
4. Run the following commands to install the next few packages:
   1. Run **pip3 install RPI.GPIO**.
   2. Run **pip3 install adafruit-blinka**.
   3. Run **pip3 install adafruit-circuitpython-gps**.
5. At this point the GPS is ready to be used. If the connection type changes, the python file at which the GPS was initialized from needs to be updated as well. Otherwise the GPS will not work as intended. Ideally this could be kept in a global settings list and updated from there, however I do not see the need to have an overly flexible design on this matter as a big deal.

**Setting up VNC Viewer**

The Raspbian OS comes preinstalled with VNC Viewer, it just needs to be setup. This will allow a developer or IT to access the Raspberry Pi wirelessly without the need to have an external monitor, keyboard, and mouse. Instructions based off of <https://www.raspberrypi.org/documentation/remote-access/vnc/>

1. Install VNC Connect by running **sudo apt-get install realvnc-server realvnc-viewer**.
2. In the Raspberry Pi menu, go to **Menu > Preferences > Raspberry Pi Configuration > Interfaces**. Make sure that VNC is enabled.
3. Find the private IP address of the Raspberry Pi. This can be done either through SSH while connected to the Raspberry Pi, or if privileged try to find the Raspberry Pi on the network. Use this IP address in the VNC Viewer application of the computer trying to remote into VNC.
4. At time of publication, the sole-SmartAVL Raspberry Pi has a password of “ece492” to connect to it. This is using a VNC password instead of the UNIX login to login into the computer. We highly recommend that this password is changed should this project be used outside of a development environment. If on a new installation, the default username is **pi** and default password is **raspberry**. The login can be changed in the VNC menu of the Raspberry Pi by opening **Menu > Options > Security** and changing the Authentication type and settings there.