WildlifeSystems - biodiversity technologies

Ed Baker

2022-08-17

Contents

About **5**
0.1 Support 5

1 Biodiversity Technologies **7**
1.1 Structure of Wildlife Systems 7

2 Sensor Networks **9**

3 Environmental Sensors **11**
3.1 How sensors work 11

4 Sensors in WildlifeSystems **13**
4.1 Sensors included in the base system 13
4.2 Installing sensor support 13
4.3 Reading data from a sensor 14
4.4 The sensor reading process 14
4.5 Installing new sensors 14

5 Implementing new sensors **15**
5.1 Reading the sensor 16
5.2 Setting the environment 17
5.3 Install 17
5.4 The installation process with `si` 17
5.5 Submitting packages to WildlifeSystems 17

6 Power Management **19**
6.1 Installation of power management tools 19
6.2 Turning funtionality on and off 19
6.3 Considerations 19

7 Developer Guidelines **21**
7.1 Documentation 21

About

This book explains the technologies developed as part of WildlifeSystems and how they can be implemented in real-world scenarios.

0.1 Support

WildlifeSystems nodes were originally developed and used as part of the Leverhulme Trust funded Automated Acoustic Observatories project at the University of York. Additional development is undertaken as part of the Urban Nature Project at the Natural History Msueum, London.

Chapter 1

Biodiversity Technologies

What are *Biodiversity Technologies*?

1.1 Structure of Wildlife Systems

1.1.1 Overall Philosophy

1.1.2 Packages

Chapter 2

Sensor Networks

Chapter 3

Environmental Sensors

3.1 How sensors work

3.1.1 Temperature

3.1.2 Humidity

3.1.3 Air Pressure

3.1.4 Gases

3.1.4.1 Heated Gas Resistance

Chapter 4

Sensors in WildlifeSystems

The WildlifeSystems platform comes with support for some popular existing environmental sensors, although there are many on the market and the range available is subject to constant change. The modular nature of WildlifeSystems allows for new sensors to be easily integrated if the need arises.

4.1 Sensors included in the base system

The Raspberry Pi does not come with environmental sensors, however there are several onboard sensors that are used to monitor the operation of the hardware, to prevent crucial components from overheating, including the temperature of the CPU and GPU chips. WildlifeSystems provides access to these sensors through the `sensor-onboard` package, as well as providing some *software sensors* that report the free memory and free SD storage available. These can be useful for detecting and resolving possible issues on a sensor node before a serious problem arises.

4.2 Installing sensor support

Support for sensors is installed as part of the node installation process, however it is possible to install the `sensor-control` abstraction layer onto any Raspberry Pi system using the command below.

```
wget -O - https://raw.githubusercontent.com/Wildlife-Systems/sensor-control/main/install | sudo b
```

This will install the `sensor-control` and `sensor-onboard` scripts into the system, as well as installing a small of number of supporting packages if they are not already installed.

4.3 Reading data from a sensor

The sensor read command, `sr`, can be used to read sensor data.

```
sr onboard
```

This will give a JSON string listing information about each sensor, and the current reading. This information can be presented in a more human readable form by piping the output to the program `jq`, a command line JSON processor.

```
sr onboard | jq
```

4.4 The sensor reading process

The sensor reading process in WildlifeSystems has five steps.

1. The `sr` command identifies which sensor script to route the request to.
2. The sensor script calls the `sc-prototype` script to obtain a template ("prototype") of the JSON request.
3. The sensor script access the relevant sensor(s) and populates the values in a template for each sensor reading, before returning a JSON array of populated readings to `sr`.
4. The `sr` script populates additional information for each reading, providing the `node_id` and a `timestamp` for each.
5. `sr` returns the finalised JSON array to the user.

4.5 Installing new sensors

Chapter 5

Implementing new sensors

New sensors should be implemented as new packages (i.e, a new GitHub repository). Packages have a standard format, and should be named `sensor-<sensor-name>` where `<sensor-name>` is short, descriptive of the sensor (e.g. model number), and unique within the WildlifeSystems ecosystem.

The structure of a basic package is given below.

```
.
+-- inst
|   +-- files that are not part of the package structure
|       used during installation (e.g. 3rd party scripts
|       to be copied to /usr/bin/)
|
+-- sensor-<sensor-name>
|     Executable to read sensor and print JSON of readings
|     (copied on install to /usr/bin/)
|
+-- <sensor-name>
|     Configuration file
|     (copied on install to /var/aao/sensors/)
|
+-- install
|     Bash file run once on install - used to install packaged
|     dependencies, etc.
```

As many people implement various sensors on the Raspberry Pi, it is likely that some sort of solution is already available, that can be tweaked to output readings in the standard format required. However, you must ensure that any licensing conditions are met. In particular, an open license is required if submitting your sensor package to WildlifeSystems for inclusion in the ecosystem.

5.1 Reading the sensor

File `sensor-<sensor-name>` provides the functionality to read the protoype JSON, populate the JSON with sensor readings, and print the output.

The file can be written in any programming or scripting language, but to prevent overhead consideration should be given to minimising the number of new packages installed.

5.1.1 In bash

```
#!/bin/bash

# Read the prototype JSON
JSON=$(sc-prototype)

echo -n "["

# Code to read the sensor value into GPU_TEMP

# Use `jq` to modify JSON
SENSOR=$(echo $JSON | jq ".sensor |= \"onboard_gpu\" | .measures |= \"temperature\" |
echo -n $SENSOR

echo "]"
```

5.1.2 In Python

```
#!/
import os
import json

# Read the prototype JSON
stream = os.popen('sc-prototype')
output = stream.read()

# Pre-populate with sensor metadata
temperature = json.loads(output)
temperature["sensor"] = "bme680_temperature"
temperature["measures"] = "temperature"
temperature["unit"] = "Celsius"

# Code to read sensor and output in variable `sensor_reading`
temperature["value"] = sensor_reading
```

```
# Output the JSON in an array
print("[",json.dumps(temperature),"]")
```

5.2 Setting the environment

The file `<sensor-name>` in the package specifies information that modify the environment of the Raspberry Pi (e.g. if the i2c interface should be enabled) before the `sensor-<sensor-name>` script is run. This allows sensors with different requirements to run sequentially on the same node.

The file must always be present, even if it is empty. During installation the file is moved to `/var/aao/sensors/` and the list of files in this directory indicates to the system which sensors are installed.

TODO: i2c

5.3 Install

The file `install` in the directory is run once, when the sensor package is installed. This allows for the installation of packages and scripts necessary for the functioning of the package.

The file is executed by `bash` and the use of `sudo` is allowed.

5.4 The installation process with `si`

The sensor install script, `si`, from `sensor-control` is used to install sensor packages. For developer reference the installation process is described below.

1. `si` clones the `Wildlife-Systems/sensor-<sensor-name>` repository from GitHub.
2. The `install` script is executed.
3. `sensor-<sensor-name` is marked as executable and move to /usr/bin/.
4. `<sensor-name>` is moved to /var/aao/sensors/.
5. The cloned repository is removed from the local filesystem.

5.5 Submitting packages to WildlifeSystems

Submitting packages (where licensing allows) to WildlifeSystems allows the ecosystem to be developed and sustained collaboratively by the user community.

Packages can be sent to the administrators as a compressed file, or a request can be sent to fork an existing GitHub repository. Contact details can be found at https://wildlife.systems/contact.html.

Chapter 6

Power Management

Power management on the Raspberry Pi is useful when deployments are made that are powered by batteries and/or renewable sources such as solar power that are intermittent.

In addition, there are small environmental benefits on consuming less power on systems which have continual grid power.

The `pi-pwr` script can be used to turn off unused functionality, either always or just when it is not required.

6.1 Installation of power management tools

The power management tools are installed as part of the node installation process, however they can be easily installed independently on any Raspberry Pi system.

```
wget -O - https://github.com/wildlife-systems/pi-pwr/raw/master/install | sudo bash
```

6.2 Turning funtionality on and off

6.3 Considerations

Disabling network functionality (WiFI / Ethernet) will prevent the node from communicating until either the functionality is turned back on or the Raspberry Pi is restarted. If disabling all connectivity is desired periodically then the functionality to turn these systems back on must be scripted.

Chapter 7

Developer Guidelines

7.1 Documentation

There are three main kinds of documentation in the WildlifeSystems project.

1. **Code comments**: Primarily for future you (or someone similar) to get to grips with your code. *Why does this code do this?*.
2. **Package documentation**: Describes what your package does, how it interacts with the larger system. *What will this package do for me?*
3. **Overall project documentation**: This manual. *How do I use the entire system for my research?*