Raspberry Pi in Wearable

COP5859 – Semantic Web Programming

Santiago Aguerrevere & Maciej Medyk

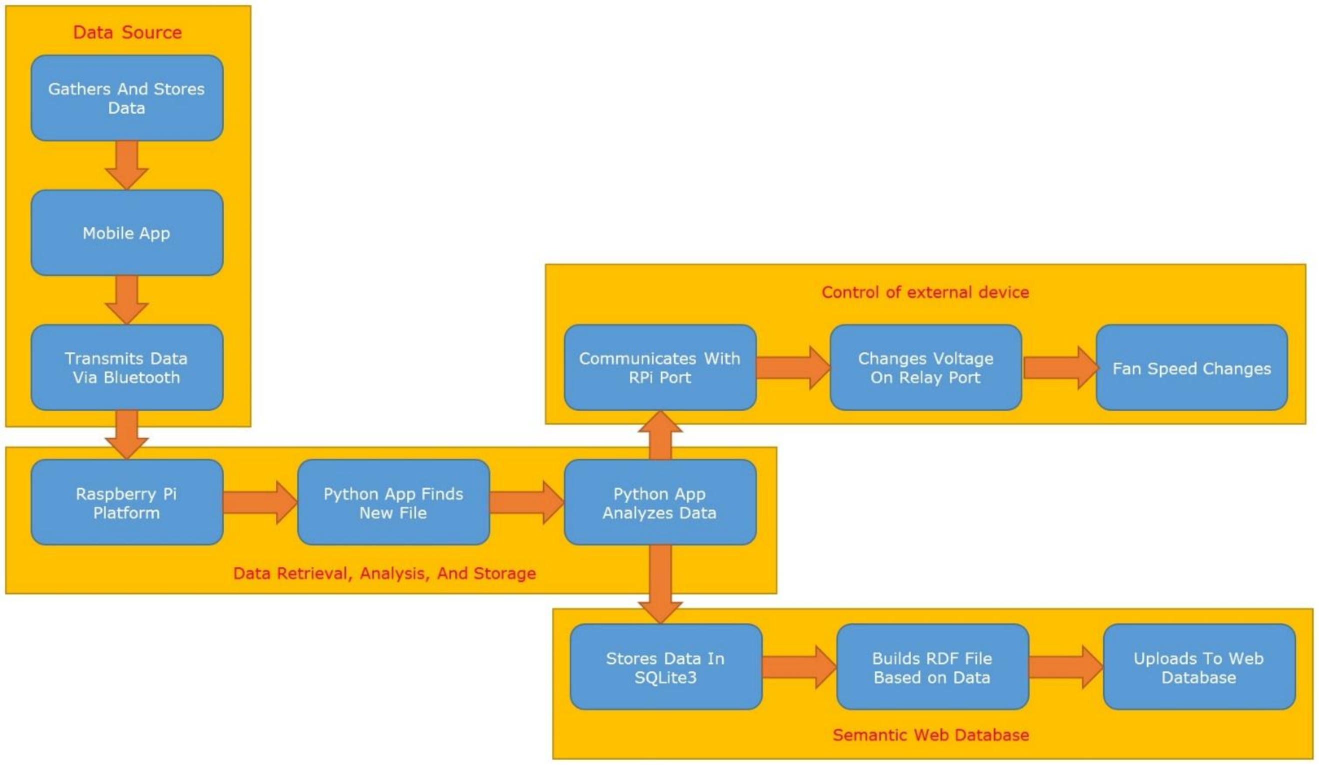
Project Overview

Raspberry Pi in a Wearable is a project in which goal is to create an environment where application on the smart phone or stand-alone pedometer communicates with Raspberry Pi and sends data for analysis. Based on of the data, Raspberry Pi then would control a device like a cooling fan by turning it on or off or adjusting its speed. Additionally, project encourages the students to expand the functionality into semantic web database that would store the information that was processed by the device. This project is suggested as one for multiple students and is publishable as a paper.

Scope of Work

The project will involve four different main modules that would make it into working environment. The first module would involve obtaining initial data that would be done via mobile app or pedometer device. In here we explored variety of options including two leading pedometer manufacturers, many iOS and Android apps that currently exist in market and we concluded that the best way to move forward would be to try to develop own simple pedometer application that would store data and send it via Bluetooth to Raspberry Pi.

Second module of the project is focused around automatic retrieving, analysis, and storage of data that was sent from the pedometer device. The idea here is to create an python based application that automatically scans for incoming new files and when detects it, then it analyzes it, sends appropriate signal to the output port and also stores the data in local database and sends the data up into the web database portal.


Data Source
Gathers And Stores Data
Mobile App
Transmits Data Via Bluetooth
Control of external device
Communicates With RPi Port
Changes Voltage On Relay Port
Fan Speed Changes
Raspberry Pi Platform
Python App Finds New File
Python App Analyzes Data
Data Retrieval, Analysis, And Storage
Stores Data In SQLite3
Builds RDF File Based on Data
Uploads To Web Database
Semantic Web Database

Third module of this project focuses on controlling the external device and requires knowledge of how port interface would turn on or off that would lead to control over the external device connected to the Raspberry Pi.

Last module would focus on storing the data and converting it to RDF format and upload that data into the web database.

Data Source (Mobile Application)

One of the biggest challenges of the project is to find the device or an application that would could use. We explored a variety of options in wearable pedometer area including two leading manufacturers Omron and FitBit. Omron devices mostly was focused on USB data transmission; however, root of the device was never allowed to be visible in either Windows or Linux environment; therefore, we moved on to testing FitBit device. From FitBit we tested model Zip that comes with its own Bluetooth dongle and allowed the device to be synced using a python based program called ‘galileo’, but again we came across an issue of encrypted file.

From iOS available applications the most promising one is called Pacer which stores the data on incremental basis of 15 minutes and has export functionality of coma separated file via email. This is possible solution but less desirable due to lack of Bluetooth connectivity, but we found out that Bluetooth is not allowed form of transmission of data on iOS devices. In research on Android we found a promising application called Pedometer+ that allows Bluetooth connectivity; however, it only stored the data on incremental basis of 1 day which would prove useless in this project. At the end the decision is made to try to develop an Android based application that would have very simple interface, gather and store data, and send it to the Raspberry Pi via Bluetooth.

Data Transmission

Bluetooth transmission would be the most feasible option as it allows the communication between devices with absence of access to the internet. In addition when file is sent from device to Raspberry Pi it is stored in ~\Downloads folder which makes a predicable file path for the program to search for incoming files.

Another benefit of this type of communication is that the device once it’s paired with the Raspberry Pi it won’t need any additional authentication. The other route to take in case we don’t succeed with development of an app is to create python code that would regularly log into email account via POP server and download the attachments. This route would be less feasible as it would require internet connectivity on Raspberry Pi at all times and program would be associated with an email account allowing less flexibility.

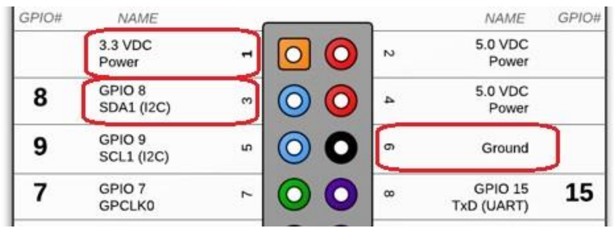
Processing Platform (Raspberry Pi and Python Application)

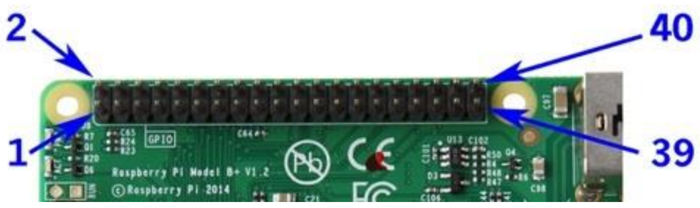
In the project we would use Raspberry Pi as hardware as it is easily modifiable to communicate with external devices and in addition it provides computational environment to run application code. The application will be based in Python and would start by executing a loop that would search for new incoming files. We expect the incoming data to be in coma separated values format; therefore, it would be easy to upload to the application using csv module.

If new file would be detected it would be processed by the code and analyzed for most recent data. If that would indicate that there is increased activity the application would communicate with Raspberry Pi Peripheral interface via use of module called ‘wiringpi2’ which allows the python code to use the 40 pin interface by increasing or decreasing voltage on certain pins. This code would be targeted towards newest version of Raspberry Pi 2 and not older versions as they come with 26 pin configuration and different board communication mode. In addition the application should be able to store data in SQLLite3 and later convert that data to RDF format and upload it to the web database portal.

Raspberry Pi Peripheral Interface

To interface with the 40 pins on Raspberry Pi we will use python and a module called ‘wiringpi2’; however, that will limit functionality only to newest Raspberry Pi 2. Using this module we will connect wires to pins 1, 3, and 6 to an Arudino relay module that would be later connect to a cooling fan. The python application based on increased activity would change the code for pin 3 to output a 3.5V charge and on decreased activity to output the charge of 0.0V. The relay then would get the voltage from those three pins where pin 1 would be Power and would go to + terminal on the relay, pin 6 would be ground and go to – terminal on the relay, and pin 3 would be a Signal and would be controlled by application and would go to S terminal on the relay. Relay would then act as a switch that would either turn the cooling fan on or off.





Python Database and Web Based Data Storage

Final module of this project would be storage of data in web database in RDF or OWL format. Python naturally works very well with SQLLite3 database which would be used to stored data locally on the device. Device would also have to take that data and convert it into RDF format and upload it to TripleStore database portal like 4store.org. We would communicate with that database via use of SPARQL module called ‘SPARQLWrapper’ that is designed for python.

Division of Work in the Project

Santiago Aguerrevere

- Develop mobile application that tracks steps
- Develop within mobile application Bluetooth method that will send data from the application to Raspberry Pi

Maciej Medyk

- Develop python code that looks for new files and processes new data
- Develop connection interface for a cooling fan and create code that controls a voltage output on the pins that will in turn control a relay

Joint Effort

- Create a solution for RDF based database that would store the results