Student Name : John Dennehy (20091408)

Project Repo URL : <https://github.com/JohnDennehy101/homeautomation>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Grade Band | Combined Knowledge | Networking Technologies | IoT Solution | Communication |
| Base |  |  |  |  |
| Good |  |  |  |  |
| Excellent |  |  | Use of IOT platforms (Thingspeak & IFTTT) ensures that application is of good prototypical standard (Automation of Light System).  Firebase ensures that data persistence occurs (Influence of weather conditions on running performance).  Integration of 3rd party services (Glitch, Strava, OpenWeather, Sonos, Google Maps, SunsetSunrise, Mapbox, Twilio) with physical computing (SenseHat, Picamera, PIR motion sensor) provides user with functionality that could be used as basis for production application. | Instruction Videos provided to describe functionality of each project component.  Visual representation of logical flow available for each project section.  Pictorial slides with block level representation of project completed and annotated with links to 3rd party services used in project.  Detailed Readme with screenshots ensures that project goals and results are clearly presented. |
| Outstanding | Programming: (Python and Javascript)  Database: (Firebase)  Web Development: (Responsive Glitch application – HTML, CSS, JavaScript)  Computer Systems: (SenseHat, PIR Motion Sensor, Breadboard, Bluetooth, PiCamera, Mini black hat hack3r, MQTT, APIs, Bluetooth) strands used in project in addition to self-acquired knowledge (Threading, use of several Python libraries, authentication and authorisation for API use – Strava). | MQTT used for first section of project (Automation of Light System) to publish data to ThingSpeak.  MatLab analysis on ThingSpeak determines IFTTT webhook that should be fired (email and Kasa smart plug actions configured).  Bluetooth and Wifi protocols used extensively in project. Range of devices used in project (Raspberry Pi, Android Phone, Garmin Smartwatch, Sonos Wifi Speaker, Kasa SmartPlug). |  |  |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Additional Comments:

**Impact of Weather Conditions on Running Performance** - API response from Strava was initially causing issues in that it was taking a few seconds to process the response (but the program was not waiting for the full process to complete). Additional research and application of Threads and events resolved the problem. Integrated Google Maps into Glitch application ([link to Glitch application](https://weather-impact-on-running-performance.glitch.me/)) and used polyline provided in Strava API response to populate and centre map with use of map markers to mark beginning and end points of latest activity. ChartJs used with Firebase realtime database data in Glitch application to provide view of run trends as well as weather conditions.

**Automation of Lighting System** - YUV image format was used with the Picamera to obtain the luminance reading to determine current room brightness (with use of numpy to load the Y (luminance) data from the stream). Also made extensive use of datetime for comparison of current time against set conditions (i.e. to compare current time with ‘dusk’ time provided in API response from SunriseSunset) as well as setting recurring time for API call each day (i.e. call out to SunriseSunset made between 09:00:00 and 09:00:25 each day).

