# JARVIS
# SMART HOME ASSISTANT

PROJECT ARTEFACT

Duy Khanh Nguyen | SIT210 – Embedded Systems Development | 29/05/2020

# Table of Contents

# I. Overview

## 1. Background

A smart home is a residence that uses internet-connected devices and sensors to enable remote monitoring and controlling of appliances, systems. Nowadays, smart home technology has grown rapidly in services that provides homeowners security, comfort, convenience, and energy efficiency by allowing users control associated smart devices using a smart home app on mobile devices via the internet. Smart home system and devices operate together, share usage data, communicate with others, and perform automated actions based on user's preferences.

## 2. Project Statement

Jarvis, a smart home assistant, is a smart system that make your home more convenient to live by connecting sensors, camera, smart appliances, light bulbs, and a GUI application for monitoring and managements. Jarvis is operating mainly at two areas which are front door area and personal room. The system provides users capabilities to monitor front door area when they are away from home, interacts with either visitors or delivery guys, monitor personal room conditions and control remotely connected appliances.

Jarvis functionalities consists of:

- Security camera
- Door message display
- Door buttons
- Automated parcel box
- Guest detection
- Automated door light
- Automation routines
- Realtime monitoring
- Realtime alerts and notifications
- Temperature & Humidity monitoring
- Automated compatible appliances
- Data analysis
- GUI application

## 3. Requirements

Jarvis is developed consists of two systems operating together which are the central processing system and the embedded system. These components are built on two primary devices which are Raspberry Pi board (central system) and Particle Argon (embedded system). The devices programming languages used in this project are Python for Raspberry Pi and similar C languages for Particle Argon. The system requirements of hardware components and software are described in detail as the following:

*Table 1: Hardware Requirements*

| Devices | Functional | Descriptions |
|---|---|---|
| Raspberry Pi 3 Model B+ | Central system CPU | Python programming, Linux OS, Wi-Fi/Bluetooth connections, 40 GPIO pins, etc. |
| Particle Argon | Embedded system CPU | GPIO pins, Particle Cloud connection, Mesh network, Wi-Fi/Bluetooth connection, Particle Web IDE, etc. |
| LCD Character Display | Door message display | GPIO/I2C connection, LCD screen, etc. |
| Noir Camera Module | Security camera | V1.3, 5MP (2592 x 1944 pixels), associated with Raspberry Pi board, etc. |
| Servo motor | Parcel box controller | Model SG90, 3.0v ~ 7.2V, 180-degree rotation, etc. |
| Ultrasonic sensor | Guest detection | Model HC-SR04, 5V, 2-400cm range distance, etc. |
| Buzzer | Guest alert | 3-30v, 80 dB, etc. |
| DHT22 Sensor | Room sensor | 5V, Temperature: -10 – $80^0C$, Humidity: 0 -100% RH, etc. |
| LED lights | Light bulbs, simulated smart appliances | |
| Breadboard, Wires | Connect devices | Male-Female, Female-Female wires, connect to Argon, etc. |
| Press buttons | Door buttons | Simple button, 8mm, etc. |

*Table 2: Software Requirements*

| Software/Package | Descriptions |
|---|---|
| Noobs/Raspbian | Raspberry Pi OS, Ubuntu, etc. |
| Thonny | Python IDE, installed on Linux, etc. |
| Python3 | V3.7, programing languages for Raspberry Pi |
| VNC (Server/Viewer) | Remote desktop display |
| PyQt5 | GUI packages for Python |
| QT Designer | UI/Layout design software on PyQt5 packages |
| OpenCV | Camera image/video rendering package |
| Threading | Threads management package |
| RPi.GPIO | Python package for Raspberry Pi general purpose I/O |
| Picamera | Mange RPi camera |
| RPLCD | LCD display controller |
| Pyparticleio | Particle cloud – Raspberry Pi communication |
| dht.h library | Particle cloud library for DHT22 sensor |
| IFTTT applets | Create triggers for notifications, managements |
| ThinkSpeak channel | Data visualization and analysis |
| Android IFTTT app | SMS message trigger on smartphone |

# II. Design Principles

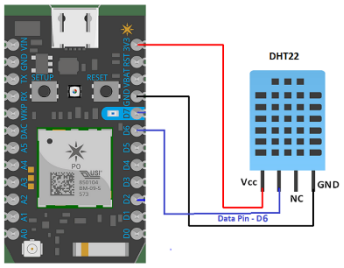
There are two parts of the system, they are the embedded system and the central system operating together and communicate wirelessly via internet. The embedded system is installed in the personal room that responsible to monitor (temperature & humidity), guest alerts, real time notifications, and data visualization & analysts. The central system is installed at the front door area that responsible to control security camera, detect guest, control lights, manage guest interfaces, host Python GUI app, control parcel box and communicate with embedded system.

## 1. Embedded System

The embedded system components consist of DHT22 sensor for monitoring, buzzer for guest alerts, LED lights for simulated appliances, breadboard and wires connected to Particle Argon for controlling using Particle Web IDE for programming. It is also connected to ThinkSpeak channel for data visualization and analysts; IFTTT applets for notifications. The details of each components are described as the following.

### a. Room monitoring

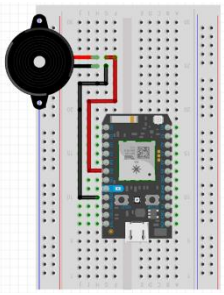
This component collects temperature and humidity data in the personal room from DHT22 sensor connected to Particle Argon as:




- Data pin: D6
- VCC: 3v3
- GND: GND

### b. Notifications

Guest alert is a buzzer noise alerts that has two alert signals for visitor and delivery:



- Visitor: 1s HIGH – 1s LOW – 1s HIGH
- Delivery; 3s HIGH

The buzzer is connected to Argon device using pin D5 as the output.

Realtime notifications are SMS messages sent to android devices using IFTTT applets and apps with contents:

If <Particle Device Name> published <Event Name>, then Send an SMS to "Phone number" with <Event Contents>.

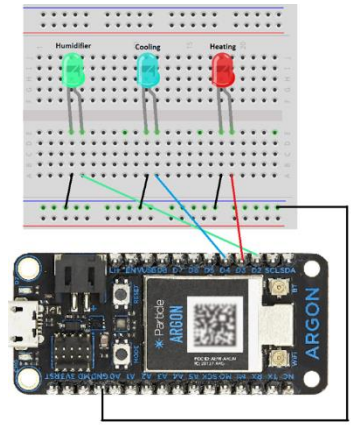
### c. Home appliances

Smart appliances in the personal room are simulated using LED lights consist of Air Conditioner and Humidifier.

- Humidifier – Green LED:
    - Connected to pin D2.

- State: either ON or OFF.

- Air Conditioner
  - Heating mode – Red LED: connected to pin D3
  - Colling mode – Blue LED: connected to pin D4.




  - State:
    - Heating: Red ON – Blue OFF
    - Cooling: Red OFF – Blue ON
    - Off: Red OFF – Blue OFF.

**d. Data visualization**

Using ThinkSpeak channel connected to Particle Cloud using integration Webhook:

- Event Name – Event Data
- API key
- Fields: Temperature, Humidity.

Collected data is visualized on 2 charts in the private view of the channel: Temperature, Humidity.

MATLAB Analysis and Data Export for further analysts.

**2. Central System**

The central system consists of LCD character display, noir camera module, LED light, press buttons, ultrasonic sensor, servo motor connected to Raspberry Pi 3 Model B+ board to control guest interactions, security camera, etc. using Python application and other supported packages.

a. Python GUI App

The app layout is designed using Qt Designer consists of several PyQt5 objects such as label, radio button, push button, text, LCD number, and so on.

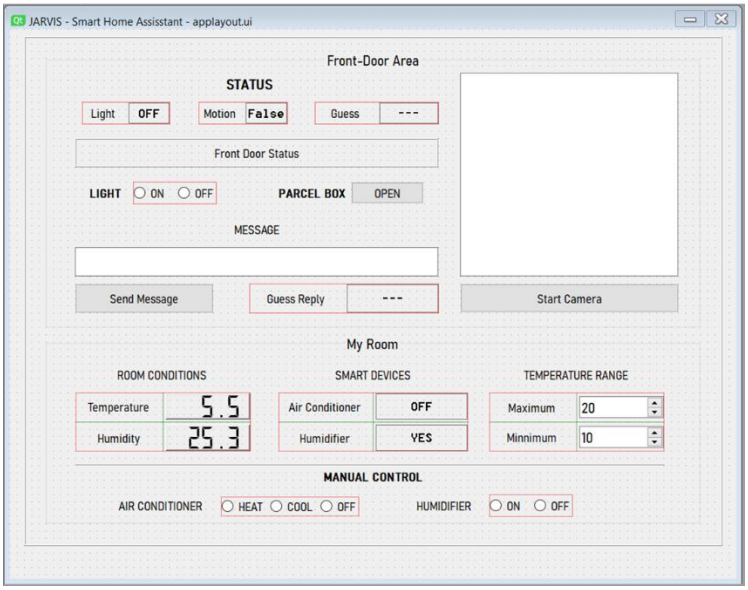



*Figure 1: Jarvis App*

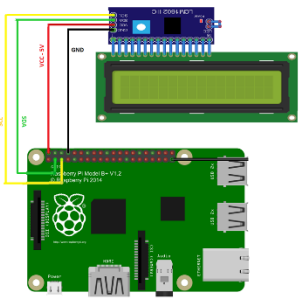
**b. Security Camera**

Noir camera board is connected to Raspberry Pi using default spot on the board. Camera feature also need to be enabled in the configuration.

The management of camera using two python packages such as picamera and opencv that display the real time streaming camera on the square camera view on the app when use press on "Start Camera" button.

**c. Guest Communication Interface**

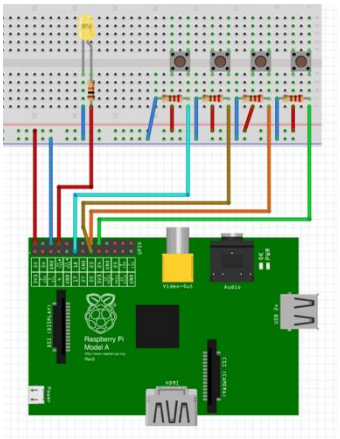
The interface consists of a LCD character display and four buttons for communications.



LCD character display is connected to Raspberry Pi suing I2C communication using PCF8574 adapter on port 0x27. It requires 4 wires such as:

- SDA, SCL
- VCC: 5V
- GND: GND

The display is controlled using RPLCD package to display user message at the front door.



Press buttons are labelled as Visitor, delivery, Yes, No as the simple interaction for guest to communicate with houseowner.

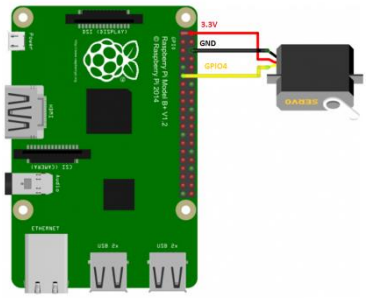
Visitor and Delivery buttons are used to inform houseowner that the guest is a visitor/delivery. The information is displayed at the "Guest" status on the app; used to make a buzzer alert in personal room and SMS message content through IFTTT.

- Visitor connected to GPIO 6.
- Delivery connected to GPIO 13.

Yes and No buttons are used to communicate with houseowner when the guest are asked for specific question. The answer is displayed at the "Guest Reply" textbox on the app.

- Visitor connected to GPIO 6 to alert user as the guest is a visitor
- Visitor connected to GPIO 6 to alert user as the guest is a visitor

**d. Parcel Box**




The parcel box is automated open/close using servo motor that connected to GPIO 4 on the raspberry pi. Servo motor is programmed using pulse with modulation (PWM) to control the actions (open/close).
If the parcel box is open, it sends a message to the LCD display to notify guest that the parcel box is opened when they can put the mail/package in.
In additional, when a guest pressed 'delivery' button on the guest interface, the parcel box is open automatically. Otherwise, it can be opened manually by click on "OPEN" button on the app.

**e. Door light**

At the front door, a LED light is installed to be turned on/off by houseowner demand or automatically when a guest is detected.

#### f. Guest detection

An ultrasonic sensor (HD-SR04) is installed at the front to measure the distance from appearing objects to the door. If the distance is lower than 10cm, then it detects that there is a guest at the front door. The information is sent immediately to control the light, displayed at 'Motion' status on the app.

The sensor is connected to raspberry pi using 4 pins:

- Echo: GPIO 24
- Trigger: GPIO 23
- VCC: 5V
- GND; GND

### 3. System Communications

The central system and embedded system communicate wirelessly using HTTP request and response on the internet. on the central system, pyparticleio package is used to establish a connection to the Particle Cloud with Argon information. Raspberry pi retrieves room data included Temperature, Humidity, Air Conditioner status, Humidifier status from the Particle Argon through variables value. Then, the result is displayed on the app to inform users room status in real time.

# III. Prototype

## 1. Architecture

### a. Embedded System

The firmware named roomcontroller.ino is programmed and uploaded to Argon device using Particle Cloud IDE. It consists of 5 parts:

- Initialization: declare variables, pins and library used in the firmware:
    - Variables: temp, hum, Temperature, Humidity, MaxTemp, MinTemp, Humidifier_status, AirConditioner_status.
    - Pins: HumidifierPin, HeaterPin, CoollerPin, Buzzer, DHTPIN.
    - Library: <Adafruit_DHT22>

- Setup() function: setup environments to use such as LED pins output, sensor dht22 input, cloud variable (Temperature, Humidity, Humidifier, AirConditioner), subscribe events (Humidifier_controller, AirConditioner_Controller, Guess_Notifications, Max_Temperature_Changed, Min_Temperature_Changed).

- Loop() function: primary infinite loop of the program that collect data from the sensor, evaluate data, publish data events (Room_Temperature, Room_Humidity) and automated controller of Temperature. This loop function is delayed every 5 seconds for processing.

- Subscribe handler functions: by registering subscribe events, each events call a specific function as response consists of humidifierController (Humidifier_controller), airConditionerController (AirConditioner_Controller), NotifyUser (Guess_Notifications), roomMaxTemperatureController (Max_Temperature_Changed), roomMinTemperatureController (Min_Temperature_Changed).

- Functions: callable functions used for specific purpose such as TemperatureController (automated controller for air conditioner), airConditioner (set mode for air conditioner).

Raspberry Pi are able to retrieve directly values from Particle cloud variables using pyparticleio package via HTTP requests & responses. it is also allowed to publish privately or subscribe events on the connected particle cloud. This connection works as communications and data exchanges between embedded system and central system.

### b. Central System

The central system, Raspberry Pi, is programmed using Python language on Thonny IDE that consists of various objects and variables to control over a number of connected devices and functions. The Python program consists of 5 threads that run continuously and responsible for different duties in the program such as:

- Main thread: run and control the window objects application using PyQt5 package. It loads the layout created from Qt Designer, register Qt5 objects for usage, display it on the screen and update data for its object every 1 second.
- ESConnectionThread: establish connection to Particle cloud, retrieve cloud variables value and store it in global variables in the program, waiting for flag to send command to particle cloud using publish events.

- DevicesThread: setup input/output pins for connected hardware devices for controlling such as servo motor, ultrasonic sensor, LED; then run continuously functions to calculate distance to ultrasonic sensor, wait for open parcel command and turn on light.
- GuestScreenThread: setup environments for LCD display and buttons for guest interactions interface. It is also run continuously to wait for displaying messages, button pressed input.
- CameraThread: declare camera and variables to stream real time camera capture at the front door on the camera view area.

As using 5 different threads, it has risen a challenge of communications between threads to exchange data and demands. To solve this problem, global variables are created in the main thread that every thread can retrieve from and upload value to it when it is declared in other threads.

## 2. Prototype codes

Link to GitHub: https://github.com/NightKing9/jarvis.git

The prototype codes consists of a Python code named Jarvis_window.py is the main program of Jarvis application, applayout,ui is the app layout created using Qt Designer, Argon code folder and a Python code to demonstrate working of camera named camera_controller.py.

## 3. Evaluation Approach

A scenario is used to test the working prototype:

- There is a guest come to the houseowner and stand at the front door.
- The ultrasonic sensor detects guest motion by calculating the distance from the object to the door.
    - If the distance < 10cm: detected is True → display on window app status.
- The guest press guest button (either visitor or delivery).
    - The buzzer alert ring in the personal room inform there is a visitor or a delivery.
    - Guest status is updated on window app.
    - SMS message is sent to android phone number.
    - LCD screen display a greeting message.
- When the guest is a delivery, open parcel box for mail/package.

- If houseowner type in question/message and send to LCD display, the guest can reply by pressing Yes/No button.
- Meanwhile, in the room area window, temperature and humidity are updated every 5 seconds and displayed on LCD number.
- There are several radio buttons that houseowner can press to control status of connected devices in central system or embedded system such as light, parcel box, air conditioner, humidifier.

Link to demonstration video where this scenario is tested on the system: https://youtu.be/w6MGwcngjZQ

# IV. User Manual

To develop Jarvis smart home assistant systems, the recommended progresses should follow the below steps:

1. Setup embedded system: connect sensor, buzzer and LEDs following the design principle for Particle Argon system. Upload and run the firmware to confirm the components are connected and working normally (data are collected correctly).
2. Create ThinkSpeak channel for data visualization.
3. Create a Python program to test connection and communication between Argon and Raspberry Pi. Retrieve and display data on shell command.
4. Connect devices to Raspberry Pi separately and testing using a Python code for each connected device:
   - LCD display using I2C
   - Buttons
   - Servo motor
   - Ultrasonic sensor
5. Create app layout using QT Designer.
6. Load and display UI layout using a Python file.
7. Setup main Thread for window app:
   - Initialize PyQt5 objects on the layout in variables for using.
   - Create control functions for app obkects.
8. Create class and functions for other threads (variables, functions and logics can be retrieved from the previous GitHub repository link):

- lcd()
- ArgonConnection()
- ControlDevices()
- Camera

9. Initialize and start threads along with main threads.
10. Create an update loop for window objects using QTimer in the main thread.
11. Run and evaluate the system with a specific scenario (follow the evaluation approach described previously).

# V. Conclusion

Jarvis – smart home assistant is a simple system to build in your smart home that provide a huge capability of improvements and customizations. It's using most of common devices and sensors in the current IoT industry such as Raspberry Pi, Particle Argon, LCD display, LEDs, Noir camera, and so on. Therefore, to achieve high quality of project and usability of system, creativity is the most important aspect in developing progress. The recommendation to reach the highest project scope is starting early on project idea as it's giving developer time to add, remove or change components; to design system architecture for easier development; to research on programming approach; and to evaluate the system prototype. As the result, IoT is one of the leading industries in the future modern world where everything is digitalized and connected, especially IoT smart home. Jarvis – smart home assistant will be an interesting hands-on project to develop and