Home Automation System

Jacob Minyoung Huh, Jene Li, Michelle Nguyen
What is it?

A home automation framework that aims to remedy the problems of current home automation systems.

- Functional despite lack of internet connectivity
- Simple to use with existing sensors in market
- Easily extendable to any custom sensors
- Easy to build applications over
General System Model
Sensors - Overview

- Low level hardware components used to read data
- Fed toward the main system in which the data is used for higher level applications
- Created libraries for sensors so that it can be easily used for applications.
Sensors - Types

1. Carbon Monoxide sensor -MQ-7
2. Combustible gas sensor -MQ-2
3. Humidity and Temperature Sensor Breakout - HIH6130
4. Infrared Proximity Breakout - VCNL4000
5. Luminosity Sensor Breakout - TSL2561
6. Triple-Axis Digital-Output Gyro Breakout - ITG-3000
7. SparkFun Barometric Pressure Sensor Breakout - BMP180
8. Ultrasonic Range Detector - LV-MaxSonar-EZ3
9. SparkFun Sound Detector - LMV324
Sensors - Communication

- Divided into 2 communication methods
- One set uses I2C protocol, where communication between sensor and processor is made through acknowledgement
- Other set communicate purely through reading analog values of the output pin of the sensor
- According to the type of communication, libraries were created/edited
Basic Model of Libraries for Pure Analog

- Read data
- Check validity
- Measure noise/ambient
- Compute gain according to noise
- Read scaled data

In all I2C sensors

In some I2C sensors
Basic Model of Libraries for Pure Analog

Read Request

Read data

Check validity

Error

Connection and validity

Default transition
True to evaluation
False to evaluation
Transition depends on the availability of the library

Measure noise/ambient

Compute gain according to noise

Read scaled data

Data value conversion

Push data
I2C Represented as State Machine

Basic Model of Libraries for I2C

- Check sensor address (Slave address)
- Check sensor product revision number
- Retrieve data
- Check validity
- Measure noise/ambient
- Compute gain according to noise
- Read scaled data

In all I2C sensors
In some I2C sensors
Basic Model of Libraries for I2C

Initialization
- Check sensor address validity (slave address)
- Check sensor product revision number validity
- Retrieve data
- Check validity
- Compute gain according to noise
- Measure noise/ambient
- Read scaled data
- Read request
- Data value conversion
- Push data
- Error
- Connection and validity

Default transition
True to evaluation
False to evaluation
Transition depends on the availability of the library
Libraries represented in State Space Model

Dependent = \{true,false\}
Rev = \{true,false,absent\}
S.A = [0,255]
Data = [0,1]
Read_Request = \{true,false\}
Valid = \{true,false\}
Data_Calculation(int Data) = R
Read_Data() = [0,1]
Find_Gain() = R

\( \neg \text{rev} \land (\text{S.A} \lor \neg \text{S.A}) \) / 
\( \text{read} \_\text{request}/ \)
\( \text{Data} = \text{Read} \_\text{Data}(); \)
\( \text{Valid} = \text{Valid} \_\text{Data(Data)}; \)
\( \text{valid} = \text{False}/ \)
\( \text{valid} \land \neg \text{dependent}/ \)
\( \text{valid} \land \text{dependent}/ \)
\( \text{gain} = \text{Find} \_\text{Gain}(); \)
\( \text{gain} \times \text{Data} \_\text{Calculation(Data)} \)
\( \text{Dependent} \)
Connection of Multiple Devices

- These libraries support multiple sensor connections to a single processor.
- I2C Sensors we are using for the project all have unique slave addresses that allows multiple connections to the main processor.
- Applications that require multiple sensors can utilize these libraries without requiring more pins.
- Abstract away frontend knowledge required to utilize multiple sensor data.
Components - Network

- Sensor readings are passed to the server using a lightweight TCP connection.
- Users can access stored data through a mobile device or desktop.
Components - Local Server

- Acts as TCP server and receives data from sensors
- Relays sensor data to the cloud when possible
- Periodically updates cloud if sensor data is received during time of no internet access
- Stores data locally on SD card to fortify against power outages and allow data access despite internet connectivity
Components - Local Server

- **Initialization** (WiFi, TCP server, SD card)
- **Wait for Client**
  - Update cloud with any unsent data
- **Retry internet connection**
- **Receive client data, update data locally**
- **Update data in cloud**

Flow:
- Initialization → Wait for Client
- Wait for Client → Update cloud with any unsent data
- Update cloud with any unsent data → Wait for Client
- Wait for Client → Retry internet connection
- Retry internet connection →互联网连接/  
  - reconnect/
  - ~internet connectivity/
- Internet connectivity/ → Receive client data, update data locally
- Receive client data, update data locally → 更新数据到本地
- Internet connectivity/ → Update data in cloud
Components - Cloud Server

- Deployed to cloud via Heroku
- Allows users to access data regardless of location
- RESTful API provides simple interface for users to easily query for data they need, making it easy to build applications upon our architecture
Future Plans

- Enclose the boards in rugged cases to improve robustness.
- Include Central Role bluetooth capability (dependent on release of S130 SoftDevices mbed support)
- Include support for uploading applications via Bluetooth