Project Proposal

Yao Yuan

[yyuan25@wustl.edu](mailto:yyuan25@wustl.eduAlex)

[Alex](mailto:yyuan25@wustl.eduAlex) Lancaster

alex.lancaster78@gmail.com

Yunfei Gu

gu.yunfei@wustl.edu

**ABSTRACT**

In this proposal, we are going to describe our IoT project.

**Project Description**

# Motivation

As IoT has been playing an increasingly important role in our daily life, we are wondering that how to better utilize these tools to improve our living qualities. One major usage is to set up a surveillance system that can monitor room temperature, moisture, etc.

# Overview

Users can monitor the room temperature via mobile apps or websites and send commands (by clicking or tapping) to the corresponding equipment to regulate room temperature.

# Architecture

The overall architecture has multiple layers (see Figure 1). Sensor 1 (temperature sensor, moisture sensor, etc) is connected to a Raspberry Pi, which served as a data collector. Whenever the data is received, the data will be published to the AWS IoT labeled as a certain topic. On an AWS EC2 instance, a lightweight application, which is subscribed to this specific topic, collects all the published message and stores them in a remote Mongo database. Meanwhile, a server is constantly polling messages from the database. After some data analysis, the result will be sent to the client side. The client side can be either a mobile app or a website. The data collected by the sensor along with the analysis result will be showed. The users can send commands back to the Meteor server. The command will be further stored in the database. The intermediate server will periodically poll the database, and once a command is recognized, the command will be received by the Raspberry Pi via AWS IoT topic subscription. Sensor 2, which is also connected to the Raspberry Pi, will receive and execute corresponding commands.

# QoS Goal

No data (sensor data) should be lost. Users are able to monitor and regulate the room temperature in a real-time fashion. The message latency (the time interval for the message to be received by the user apps) should be less than 1 second.

# ../Dropbox/Apps/drawio/CSE520%20Project.pngPlan & Milestones

**By 10/10:**

Finish the basic setup of the overall architecture and complete configuring sensors.

**By 11/14:**

Finish developing the intermediate servers and meteor servers.

Work on the communication protocols between user applications and backend servers.

Might add new features.

**By 12/5:**

Finish the project.

# Responsibilities

**Yao Yuan:**

Design and develop intermediate server and meteor servers. Construct the overall architecture.

**Yunfei Gu:**

Establish the connection between sensors and Raspberry PI. Handle hardware problems.

**Alex Lancaster:**

Figure 1. Overall Architecture

.

Optimize the user application, including UI, protocols, etc.