Smart Home Air Filtering System: A Randomized Controlled Trial for Performance Evaluation

Kyeong T. Min[a], Philip Lundrigan[b], Katherine Sward[c], Scott C. Collingwood[d], Neal Patwari[a]

[a]*Electrical and Computer Engineering, University of Utah, United States*
[b]*School of Computing, University of Utah, United States*
[c]*College of Nursing, University of Utah, United States*
[d]*School of Medicine, University of Utah, United States*

Abstract

Airborne particulate matter (PM) exposure exacerbates asthma and other respiratory and cardiovascular conditions. Using an indoor air purifier or furnace fan can reduce the PM concentration, however, these devices consume significant energy. We designed and built an air quality automation system called SmartAir that uses measurements from PM sensors to control a home's furnace fan. When PM is high, SmartAir turns on the furnace fan, pulling air through a filter to reduce the air pollutant concentration. When the PM is low, the fan is off to conserve energy. We describe an architecture we introduce to automatically perform a repeated measurement randomized controlled trial (RCT) to evaluate SmartAir. We argue this is an appropriate scientific method to use to evaluate a variety of IoT systems that purport to improve our living conditions but whose performance is complicated by individual differences and confounding variables. We deployed SmartAir in four homes for 350 days in which each day has a randomly chosen experimental condition. The results demonstrate that SmartAir achieves air quality approximately as good as when the fan is always on (average $PM_{2.5} = 6.13$ $\mu g/m^3$ SmartAir vs. 5.71 $\mu g/m^3$ On) while using 58% less energy. SmartAir also finds statistically significant lowering of $PM_{2.5}$ for SmartAir vs. normal furnace fan operation. Although SmartAir increases fan on percentage, its lowering of $PM_{2.5}$ can not be achieved simply by turning on the fan more often without an air quality sensor.

Keywords: Indoor Air Quality, Randomized Controlled Trial, Internet of Things (IoT), Smart Health

1. Introduction

Increased exposure to airborne particulate pollution can trigger asthma exacerbation [1] and is associated with increased risk of lung and heart disease and shorter life expectancy [2], approximately 1.5 years of life lost per each additional 10 $\mu g/m^3$ in $PM_{2.5}$, that is, the concentration of particles less than 2.5 μm in diameter. Although this effect is a population average – some people's lifetime will not be affected – people whose lives are shortened may die 9-11 years earlier than if they had been exposed to $PM_{2.5}$ at a level 10 $\mu g/m^3$ lower [3]. People in the US spend approximately 88% of their day indoors [4], thus indoor air pollution is a major influence on health.

Using an air purifier or furnace fan can filter the particulate matter out of the air. However, the cost of using the furnace fan to clean a home's air is increased energy consumption. At one house in our study, we measured that running the furnace fan continuously, 24 hours per day, consumes 273 kWh (about US$33) per month, a very considerable use of energy. It would be counter to the goal of better overall air quality, when most worldwide electric power is generated from burning fossil fuels [5], to recommend all households set their fan to always on. Yet households with people with asthma and other respiratory health issues are recommended to use air purifiers and to set their thermostat fan to be always on with a disposable high-efficiency MERV filter [6] to reduce risks to their health [7, 6].

Email addresses: kyeong.min@utah.edu (Kyeong T. Min), philipbl@cs.utah.edu (Philip Lundrigan), kathy.sward@nurs.utah.edu (Katherine Sward), scott.collingwood@hsc.utah.edu (Scott C. Collingwood), npatwari@ece.utah.edu (Neal Patwari)

In this paper, we present an air quality automation system called SmartAir that uses measurements from a PM sensor to control a home's furnace fan, pulling air through a filter to reduce the air pollutant concentration whenever the pollution level is high. When PM levels are low, the fan is off to conserve energy. With SmartAir, we hope to provide the health benefits of air purification without a major increase in electricity consumption.

SmartAir is similar to other internet-of-things (IoT) and cyber-physical systems (CPS) which purport to improve our living environments and improve our health [8, 9, 10]. As their performance is confounded with many other variables about the weather, our daily activities, and the actions of other people and systems, determining their average performance is challenging. One method of performance evaluation used in IoT research is via simulation models [11, 9]. The motivation for simulation is to run two algorithms in exactly the same conditions, which is impossible in the real world. Furthermore, models ignore real-world complexities which may affect the results. Another method is to deploy the two algorithms at different time periods in the same building. However, the weather or other temporal differences between the different periods are confounding variables which also impact the results in unknown ways, or the way the time periods are chosen may introduce bias [10]. A variation would be to deploy two different algorithms in two different buildings and run them simultaneously; however, then the structural and occupant characteristics of the two particular buildings confound the results. In fact, we observe very different PM levels and characteristics in the four homes we use for our measurements.

In this paper, we propose to advance the ability of researchers to conduct a repeated measurement randomized controlled trial (RCT) for IoT-based smart health applications. We call this RCT-automation system "thing-enabled self-science" (TESS), because it generally allows a person to conduct any repeated measurement RCT on themselves using their own IoT devices. TESS can connect to various sensors and actuators. It conducts a repeated measurement RCT of an IoT sensor/actuator system automatically. The randomization and control performed by TESS is designed to reduce bias and avoid the influence of confounding variables, and thus help researchers to make a scientifically valid conclusion. We use TESS to conduct a RCT on SmartAir, and in the end, reliably determine if SmartAir results in statistically significant differences in air quality and/or energy consumption. We use TESS to run an RCT in four homes, over a total of 350 days. Each day, TESS randomly chooses to run SmartAir or a different baseline fan operation condition.

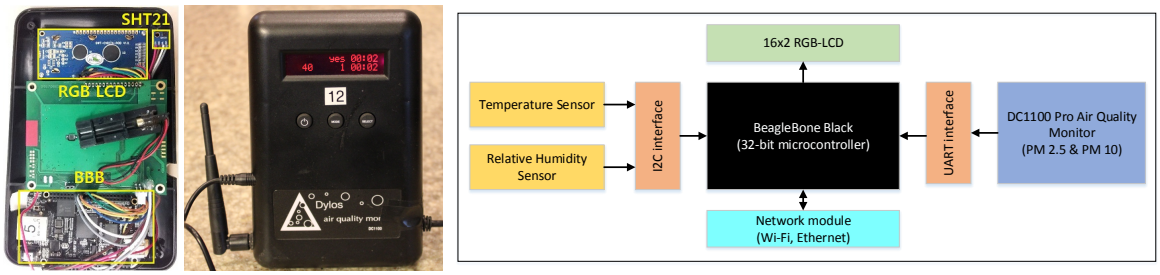
We summarize our contributions as follows:

1. We present SmartAir, an air quality automation system. We design our system to use measurements from PM sensors to automatically control a home's furnace fan to reduce the air pollutant concentration and provide the health benefits while conserving energy.
2. We present TESS, a general software framework and the RCT-automation system for researchers to conduct scientific experiments on IoT systems and make a scientifically valid conclusion.
3. This paper presents the first rigorous performance analysis of a system which filters the air using the furnace fan whenever the indoor air pollution level is high. Our experiment results show that SmartAir significantly reduces PM2.5 compared to normal furnace fan operation (6.13 vs. 8.85 μg/m^3) to a level close to when the fan is always on (5.71 μg/m^3). The fan on percentage increases in SmartAir compared to normal operation (42% vs. 24%), but is still much less then the fan is always on (100%). In other words, SmartAir can keep air pollution low in a home without leaving the fan on continuously.

The remainder of this paper is organized as follows. First, we present SmartAir, an air quality automation system in Section 2. Then, we present our general software framework called TESS and its design in Section 3. In Section 4, we describe the experimental deployments and show our results for the deployments in Section 5. We point out some important considerations for the system design in Section 6. Then, we discuss the related work in Section 7. Finally, we draw our conclusion in Section 8.

2. SmartAir

SmartAir is an air quality automation system that consists of PM sensors, a smart thermostat, and an algorithm which controls the interaction of the two. SmartAir measures indoor air pollution and reduces indoor pollutant levels by turning on the furnace fan only when air quality is poor. In comparison to operating the fan continuously, SmartAir saves energy.



(a) (b) (c)

Figure 1: (a) A Beaglebone Black w/ WiFi, RGB-LCD, and SHT21 temperature and humidity sensor are added to the inside of a Dylos DC1100; (b) outside view; and (c) hardware block diagram.

2.1. SmartAir Sensors

Here, we describe the primary sensor used in SmartAir. Multiple low-cost air quality sensors are sold for personal or home use. Each sensor has a particular non-linear relationship between actual and measured $PM_{2.5}$, which varies with temperature and humidity. Unfortunately, each pollutant has a different relationship [12] which requires extensive lab-based characterization. We worked with environmental health scientists who are focused on the reduction of pediatric asthma. To be able to contribute to their field, we chose to use the Dylos DC1100 pro particulate matter sensor, which is extensively reported on in the environmental health literature (e.g., [12, 13]). Although it is popular and well-characterized, it does not have a wireless interface or a display that is useful to non-experts. Wireless is required for real-time actuation and ease of deployment. Control of the display can be useful to answer questions about how knowing the air quality influences the behaviour of home residents. We also wanted independent temperature and humidity sensing.

We chose to create the Utah-modified Dylos sensor (UMDS) [14] to address these limitations and to provide an example in which a sensor without wireless communications capabilities is modified for use with SmartAir. The MDS adds to the DC1100 an RGB LCD screen, temperature and humidity sensors, and a TI Beaglebone Black (BBB) processor, as shown in Fig. 1. The processor runs four modules: sensing, presentation, data persistence, and communication. The sensing module reads $PM_{2.5}$, PM_{10}, temperature, and humidity sensor values. The presentation module can be configured to display alerts, sensor readings, or system health status, depending on the study requirement. Sensing data is first stored locally, and then transferred to the gateway via WiFi. The communication module is configurable, but in SmartAir uses a protocol based on the constrained application protocol (CoAP) with additional acknowledgements. Code for the UMDS sensor is publicly available [15].

After the MDS sensors are built, to verify and validate the devices, several MDS sensors are tested and compared with high-end air quality devices, such as Grimm 1.109 Portable Aerosol Spectrometer (PAS) (0.2 - 20μm in 30 size channels) [16] and traditional gravimetric IMPACT Sampler (SKC $PM_{2.5}$) [17] in a test chamber.

2.2. SmartAir Actuator

For the actuator in SmartAir, we chose the Ecobee smart thermostat. The Ecobee cloud server exposes an API to monitor and control the device. Although local control is not possible, our SmartAir system uses local data on the gateway to trigger control of the furnace fan via the cloud interface. As long as the gateway is internet connected, the lack of local control is not a problem.

A note about the terminology. In a forced-air HVAC system, the furnace fan pulls air from the return air vents, through a furnace air filter, cools or heats the air depending on the setting, and then pushes the air back out to the home. The furnace fan is always on when the system is actively heating or cooling, which we refer to as normal operation. However, the furnace fan can additionally set to be on all of the time, even when the furnace is not heating or cooling, which we refer to as “On”. SmartAir operates in between these two, leaving the heating and cooling unchanged, but additionally turning on the furnace fan when the air quality is poor regardless of the heating or cooling state.

While SmartAir could also control a stand-alone air purifier, we chose to control the furnace fan because of it has high airflow and cleans air across an entire home, and because most homes already have one.

Note that the Ecobee also provides SmartAir with a form of sensing, that is, knowing what the fan state is (on or off). SmartAir queries the Ecobee cloud server and records this value, which is later used to understand the energy consumption of the fan as a function of the condition.

Figure 2: Industrial Hygienists compared the modified Dylos units against primary and secondary sampling systems in a test chamber using known aerosols. Comparators were: A Grimm 1.109 Portable Aerosol Spectrometer (PAS) and traditional gravimetric IMPACT Sampler (SKC $PM_{2.5}$)

2.3. SmartAir Conditions

In total, we define three conditions in SmartAir for our randomized controlled trial of SmartAir. RCTs, by definition, have a "control", that is, time periods which use what would be considered a default setting. For the study of a smart use of the furnace fan, the control should be what people would normally do with their furnace fan. We imagine two different controls for our furnace settings: Normal or On. Most people use their furnace fan only when the air conditioning or heating is on. This is what we call the "Normal" condition. Some families, particularly those with people with asthma, run their furnace fan all of the time, which we call the "On" condition.

In this paper, we propose SmartAir, which becomes our third condition. In the SmartAir system, the fan is controlled as a function of the air quality measurements. It uses the maximum of any $PM_{2.5}$ reading in a two-minute window. When the maximum $PM_{2.5}$ exceeds a high threshold, SmartAir forces the fan to the on state; and only goes back to the fan off state when the maximum $PM_{2.5}$ falls below a low threshold.

In summary, SmartAir randomly controls the condition among this list of possible conditions: Normal, On, or SmartAir.

Randomization: Every night at midnight, the system randomly chooses a condition from among the three in the list (Normal, On, and SmartAir), and starts the selected condition. The condition lasts for the experiment period, set to 24 hours. The random scheduler process repeats every night at midnight. The start time and duration can certainly be customized, but we felt due to the 24-hour periodic nature of the activities and presence in a home, that a single day should be considered as a single period.

Finally we note that, for all conditions, we added a safety condition that turns off the furnace fan in case of measurements which might indicate a fire. As fire smoke can cause people to lose consciousness from smoke inhalation, spreading smoke to all rooms of a home could be deadly. We trigger a shutoff of the furnace fan when there is a combination of high temperature and extremely high $PM_{2.5}$.

2.4. SmartAir Performance Testing

To verify the correct operation of SmartAir, we ran the system in a test home to verify that the system fulfills its designed purpose: automate the furnace fan when the high pollution is detected and reduce indoor air pollution in real-time based on the defined settings. We show a segment of test data in Figure 3. The thermostat fan immediately turns on when the Sensor 12 (Kitchen) detects air pollution when the $PM_{2.5}$ level goes up. The thermostat fan turns off after the $PM_{2.5}$ is reduced below the lower threshold defined in the policy setting. We also verified that, over a several day test, that the temperature rules set by the resident were, in fact, followed by the Ecobee smart thermostat.

2.5. SmartAir Analysis

We are interested in the mean furnace fan on percentage, as well as the mean $PM_{2.5}$ level in the air, and comparisons between the two control conditions and the SmartAir condition. A generic hypothesis test for this RCT is:

H_0: There is no difference in the mean value of X between conditions A and B;

H_1: There is a difference in the mean value of X between conditions A and B,

where X is either $PM_{2.5}$ or fan on percentage, while A and B are two conditions from the set {Normal, On, SmartAir}. We do not, in general, know the standard deviation of either distribution, so our statistical analysis uses the two-sample T-test.

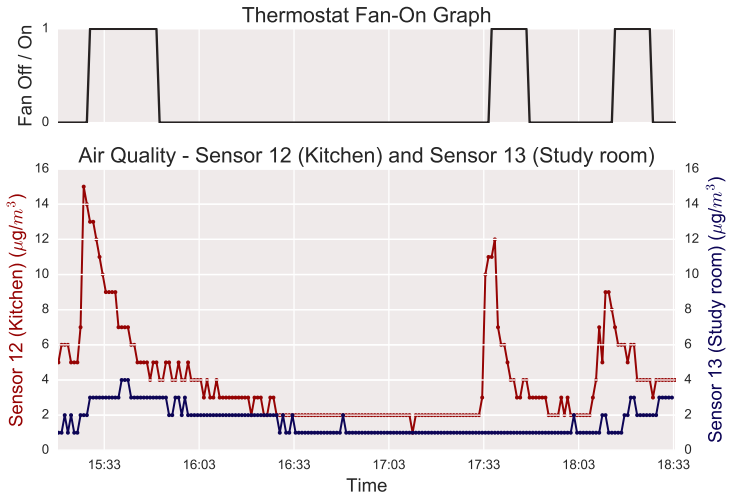


Figure 3: (Bottom) $PM_{2.5}$ sensor values in kitchen (red) and study room (blue); and (Top) furnace fan state. The fan is triggered by, and then helps to reduce, the high pollutant level in the kitchen, but pollution marginally increases in the study room.

3. Thing-Enabled Self-Science (TESS)

TESS is a general software framework for conducting a repeated measurement RCT for IoT-based smart health applications, with code publicly available at [15]. Fundamentally, TESS operates experiments. A single experiment has an *experiment period*, how long the condition is left constant. The *condition* is the particular algorithm or setting used for the actuation during an experiment period. There are at least two conditions, and TESS operates the randomization, selecting randomly and independently at the start of each experiment period which condition will be run. During the experiment period, TESS collects sensor data. Some sensor data is required to control the actuator (depending on the condition) and some is the dependent variable which is to be recorded and associated with the condition, which is the independent variable.

TESS is designed for two different types of users with differing requirements. Our system must be flexible to serve an individual who wants to evaluate an automation system in their own home[1]. Second, our system must be able to serve researchers who want to evaluate the system operating on multiple people in several homes. To meet these different requirements, TESS software framework includes:

1. Ease of RCT Setup: Minimal programming should be expected from users to be able to set up their own experiment.
2. Flexibility with IoT devices: To be widely useful, TESS must be able to communicate with a wide variety of sensors and actuators, with different communication interfaces and protocols.
3. Real time operation: Automation / control algorithms need access to the current sensor values and control the actuator, thus there should be minimal latency.
4. Automatic data processing: TESS enables automatic statistical analysis after each randomized experiment to help researchers and exposure scientists.

We describe next how the TESS is designed to address these requirements.

3.1. TESS Design

[1] Although we use "home", the system could be deployed in a workplace or other space.

Figure 4 shows the component design of the TESS gateway, which runs on a Raspberry Pi. TESS is built upon EpiFi, a general-purpose architecture designed for flexible and reliable home networks which measure and act upon exposure measurements [18]. While our Pi has Ethernet, BLE and WiFi built in, we can also connect other network interfaces (e.g., Z-Wave, LoRa) via USB or cape. Sensors and actuators are directly controlled by Home Assistant (HA) [19], which serves as a event-driven data bus for connected devices. HA is an active open source project with support for over 900 *components*, which include commonly used IoT sensors, actuators, and services [19, 20]. TESS performs device discovery and management, finding sensors and actuators that broadcast their presence and capabilities. This layer ensures that the system recovers from a device failure when a device is replaced. TESS obtains and stores data from the sensors and actuators in a local database, and provides a data service layer in which data can be operated on and aggregated as needed for actuation and later analysis. In the case of direct local control, the local database enables algorithms to query recent data. In the case in which others (patient, researchers, or medical professionals) need remote access to the view the data, the local database is transmitted over a connection encrypted by transport layer security (TLS) to a remote (HIPAA compliant) database. An acknowledgement-based transfer protocol is used to ensure no loss of data due to packet loss, internet outages, or cloud server downtime. TESS operates an *automation controller* which selects, at the start of each experiment period, a current condition. TESS operates an *actuator engine* which evaluates sensor data and controls the actuator as defined for the current condition, for example, turning the fan on whenever the $PM_{2.5}$ is higher than a threshold. When an experiment period is finished, TESS returns the actuator state to what it was previously, summarizes the dependent variable data, and stores the summary in the database for future analysis.

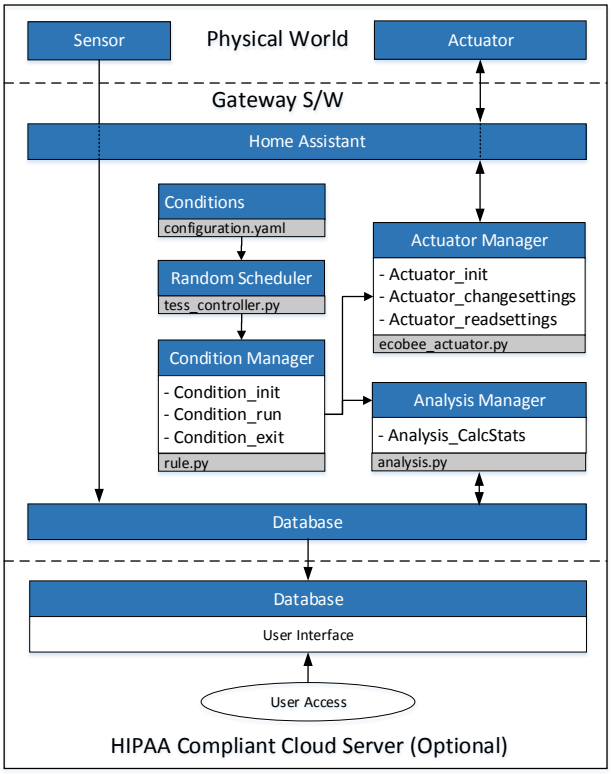


Figure 4: Component Diagram of TESS

3.2. TESS Implementation

Instead of embedding the system and experiment settings in the program code, TESS is implemented to read component settings from an external configuration file and then create their corresponding in-memory objects to update experimental conditions, monitor and control an actuator device, and run a summary analysis. This enables updating system behaviors and state changes without code changes. Furthermore, it allows TESS to easily utilize different types of sensors and actuators by updating the configuration file.

To handle diverse requirements of applications for experimental conditions, TESS uses the user's policy-based settings to control automation devices to decide if they should be actuated or not. In its simplest operation, the actuator engine sends a ON or OFF command to the smart external device to control its operation when the sensor readings for the physical conditions are above the threshold settings defined in the user's policy settings. These settings can be customized and personalized for the particular home, for example, for studying the trade-off between energy consumption and indoor air quality for an individual household.

The core components of TESS are Conditions, Random Scheduler, Condition Manager, Actuator Manager, and Analysis Manager.

3.2.1. Conditions

RCTs can have different algorithms or settings (conditions) used for the actuation during an experiment period. Instead of hard-coding the module, we use a dynamic module loading method that allows the system to be flexible for various experimental conditions. Each condition can be in active or inactive by change the active key (1: active, 0: inactive). The active means the condition is included in the random scheduling process.

TESS loads the module and create the class instance using the settings defined in the external configuration. These class instances are selected by the random scheduler process and run during the experimental period. For example, the OnCondition and SmartAirCondition objects can be instantiated from the example conditions shown below.

```
random_scheduler:
  statement: schedule.every().day.
             at("00:00").do(self.start)

conditions:
  - name: on
    mode: 'on'
    modulename: 'tess.conditions.on'
    classname: 'OnCondition'
    active: 1

  - name: smartair
    mode: 'smartair'
    modulename: 'tess.conditions.smartair'
    classname: 'SmartAirCondition'
    active: 1
```

3.2.2. Random Scheduler

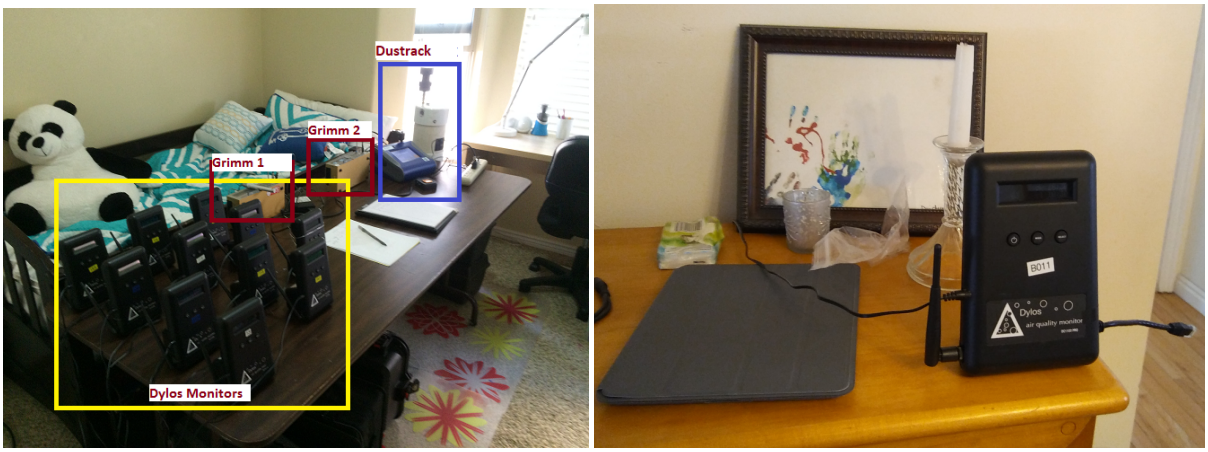
To accommodate different experimental run-time schedules, TESS reads the experiment schedule setting from the external configuration file. This approach allows TESS to meet different random scheduling needs. The random scheduler algorithm chooses a condition such as On, Normal, and SmartAir conditions, and schedule time for the next experiment as configured in the external configuration file. For example, the random_scheduler decides when (everyday), what time (12:00 AM), and which function (self.start()) to start for the experiment.

3.2.3. Condition Manager

Each condition has its corresponding condition manager, which exposes three common interfaces, such as init(), run(), and exit(). The init function initializes default values for the condition object such as its initial state and its connected actuator. The run function periodically inspects the status of the actuator and controls the actuator when threshold rules are met. When the random scheduler decides a new condition, TESS calls the exit function of the condition object, which gives the system the opportunity to summarize the data and record summary information into the database through the analysis manager. The component interaction diagram is shown in Figure 4.

3.2.4. Actuator Manager

TESS provides the BaseActuator class and allows custom actuators to extend the base functionality and properties. The basic role of the actuator manager is: connect and control an actuator device through general-purpose input/output (GPIO) pins or software application program interfaces (API). It has three common functions: init(), changesettings(id, name, value), and readsettings(id, name). These functions allow the system to change settings and read settings for the actuator. The "id" parameter is used to connect to a specific actuator if multiple actuators are connected to TESS. The "name" parameter is a unique key that is used to update or read a setting of the actuator. A new actuator manager can be extended by inheriting the base actuator (shown above), and connected using the actuator manager configuration information (shown below) because TESS dynamically loads the module and class from the configuration file.



(a) (b)

Figure 5: (a) Air quality sensors are placed in a room and calibrated with the high-end air quality devices such as GRIMM devices and a Dustrack, and (b) each air quality sensor gets deployed in a room for monitoring home environment.

```
actuator_manager:
  modulename: 'tess.actuators.thermostats.
                ecobee_actuator'
  classname: 'EcobeeActuator'
  actuator_id: 1
  home_id: tess-gateway
```

3.2.5. Cloud Server

The cloud server in TESS consists of two major components: database and user interface as shown in Figure 4. TESS is designed to run RCTs on the gateway itself with or without the cloud server. The cloud server can be connected to TESS by updating the database connection setting in the configuration file and used to store and secure the RCT data in the secure database for future analysis and user interface. When an RCT for a US-based human subject study is evaluated in TESS, it is important to consider HIPAA-compliance. TESS uses a time-series database called InfluxDB [21] for handling time-series data such as sensor data containing timestamps for temporal importance. For our user interface, we use Grafana [22], an open platform for time series analytics. The platform is readily available for various operating systems and provides visualization infrastructure for diverse graphs that can easily be configured.

4. Experiments

In this section, we describe the experimental deployments in which we conduct repeated measurement RCTs to evaluate SmartAir.

We used TESS to run an RCT in four households. Each house has a central HVAC system with the Ecobee smart thermostat. We install in each home's furnace a filter with a minimum efficiency reporting value (MERV) rating [23] of 14. This rating means that the filter is rated to remove at least 90% of the particles 1.0 to 3.0 μm in size, with somewhat lower percentages for particles smaller than 1.0 μm. While this is not as efficient at removal as a HEPA-rated furnace filter, it is significantly lower in cost and has wider availability.

The TESS Gateway is directly connected to the participant's home router via an Ethernet connection. We place two air quality sensors inside the house, one in the family room and one in a bedroom. Given a limited budget for MDS devices, we want to measure rooms in which a person spends the most time, and the family room/kitchen and the bedroom are often suggested in the indoor environmental health literature (e.g. [24]). Air quality varies by room

and over time, and we would want to actuate the furnace fan if the air was poor anywhere that a person is likely to be. Thus the maximum of the $PM_{2.5}$ values measured by the two sensors is used to decide if the furnace fan should be turned on or off in each of our SmartAir experiments, although we describe in Section 2.1 the implications of using only one sensor per house. These sensors connect to the home WiFi network to communicate with the TESS Gateway for sensor discovery, sensor configuration, and data communication, as described in Section 3.1. The MDS are set to collect one measurement per minute.

Part of the motivation for SmartAir is that individuals and their homes are different, and what may work for one may not work for another. To evaluate the variability between the performance of SmartAir in different houses, we have deployed it nearly simultaneously in four different single-family houses:

- Deployment I is a two-story house with two adults, three children, and one dog living in the house. One of the residents is generally home during the day. The house is within 100m of an interstate highway.
- Deployment II is a two-story house with two adults and two dogs living in the house. Both adults are away from the home during work hours.
- Deployment III is a two-story house with two adults living in the house. Both adults are away from the home during work hours.
- Deployment IV is a house constructed in 1963. There are two adults, two children, and two dogs. All are away from the home during work hours.

Deployments I, II, and III have people with asthma living in the home, while Deployment IV does not. Figure 5 shows an example picture of sensors being calibrated and deployed in Deployment I.

Table 1: Results for On, Normal, and SmartAir with SmartAir RCT (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$)

	Fan Mode	Days	Avg. (μg/m^3)	Std. Dev. (μg/m^3)	Fan On %	p-value vs. On	p-value vs. SmartAir	% Days above EPA
Depl. I	Normal	36	11.81	8.42	16	0.009**	0.019*	27
	On	37	7.10	5.32	100	-	0.55	11
	SmartAir	41	7.90	5.20	29	-	-	15
Depl. II	Normal	25	10.48	11.30	31	0.18	0.48	24
	On	29	6.54	8.96	100	-	0.45	17
	SmartAir	42	8.96	11.75	60	-	-	17
Depl. III	Normal	27	4.60	3.65	39	0.56	0.27	7
	On	33	3.97	4.53	100	-	0.78	3
	SmartAir	34	3.73	1.87	47	-	-	0
Depl. IV	Normal	14	6.52	4.16	7	0.23	0.85	7
	On	17	4.62	4.27	100	-	0.26	12
	SmartAir	15	6.25	3.67	12	-	-	13
Overall	Normal	102	8.85	8.36	24	0.0025**	0.0099**	19
	On	116	5.71	6.20	100	-	0.621	10
	SmartAir	132	6.13	7.29	42	-	-	10

Table 2: Runtime 20 / 40 condition results (Depl. I)

Fan Mode	Days	Average (μg/m^3)	STD (μg/m^3)
`Runtime 20`	12	11.24	9.57
`Runtime 40`	15	10.10	10.63

5. Results

With TESS randomly setting the condition each day, we have collected experimental data from the four house deployments over a total of 350 days to quantify the performance of SmartAir. We also validate the correct operation of TESS as a means to run the randomized controlled trial.

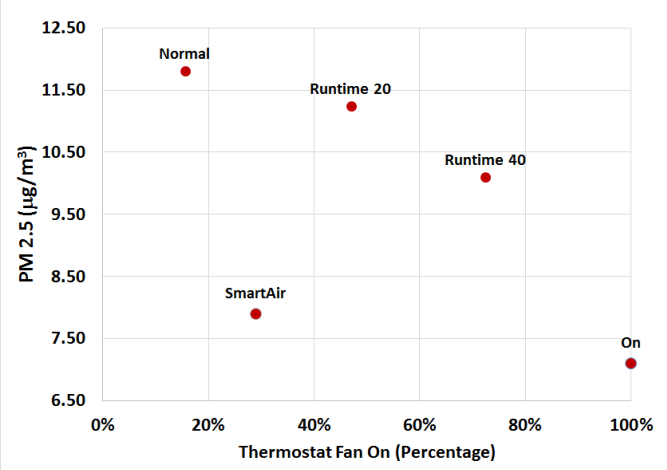


Figure 6: Average $PM_{2.5}$ and average fan on % vs. condition for Depl. I.

5.1. Summary

Table 1 shows the summary of each deployment: 1) the number of days for each fan mode, the average $PM_{2.5}$ exposure, a standard deviation (STD), 2) the average fan on percentage, 3) p-values of the T-tests which compare On, Normal, and SmartAir conditions, and 4) the percentages of days when the average $PM_{2.5}$ in the home was above the EPA standard of 12.0 μg/m^3 [25].

We look first at the data from Deployment I, our longest deployment. In our random experiments in this house, SmartAir reduces fan on percentage by 71% compared to the On condition. The average $PM_{2.5}$ is 0.80 μg/m^3 higher in SmartAir vs. On, which is a 10% difference, and is not statistically significant. However, compared to the Normal condition, SmartAir displays only 13% fan on percentage increase and achieved about 33% decrease in air pollution (Normal: 11.8, SmartAir: 7.90 μg/m^3). This air pollution decrease is statistically significant, with a p-value of 0.019. The p-value is the probability that the observed difference in mean or worse would have been observed even if H_0 (identical means) was true. Thus the low p-value quantifies that the mean $PM_{2.5}$ in the SmartAir condition is lower than the mean in the Normal condition.

The results in Table 1 show that 27% of Normal days went above the EPA standard, compared to 11% in the On and 15% in the SmartAir condition.

In summary, Deployment I is a house which benefits strongly from use of SmartAir, with air pollution reduced by one-third while increasing the fan on percentage by 13% compared to Normal operation.

5.2. Other Deployments

Three other deployments operated for fewer days, but demonstrate other aspects of SmartAir's performance. Deployments II, III, and IV each showed that SmartAir improved air quality, on average. These $PM_{2.5}$ reductions are 14.5%, 19%, and 4%, in Deployments II, III, and IV, respectively. Note that average $PM_{2.5}$ is actually lower for SmartAir than for the On condition in Deployment III as shown in Fig. 7, but the difference was not statistically significant.

Despite SmartAir improving the air on average, there were insufficient experiment days to reach statistical significance in these three deployments. Individually, these RCTs would have been unable to show that SmartAir demonstrated statistically significant improvement in air quality in these homes. Some factors may preclude SmartAir from having as large of a significance: Deployment II had a very high standard deviation of daily $PM_{2.5}$ of almost 12

μg/m^3. High standard deviation decreases the T-score and makes the same change in mean less statistically significant. Deployment III had very clean air already; its residents might already be aware of how to keep their air clean, and the numerical decrease possible with SmartAir may be relatively small. Deployment IV was a home in which windows were open many hours per day, and that may prevent SmartAir from cleaning the air faster than it enters in from the outside.

In summary, although results indicate that SmartAir is generally useful, SmartAir has different performance characteristics in each house. SmartAir provides the means to automate a scientific study, via randomized controlled trial (RCT), to validate the performance for an individual household.

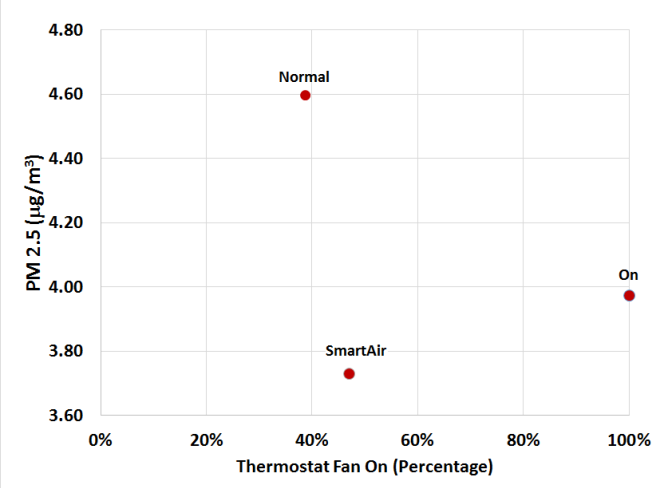


Figure 7: $PM_{2.5}$ and fan on percentage vs. condition for Depl. III

5.3. *Overall Performance*

However, with sufficient numbers of deployments across different households, we can aggregate experimental results and find out if the effect of SmartAir is significant across the general population.

Over all experiments and deployments SmartAir ran 350 days of random experiments, collecting more than 5×10^5 data points in the SmartAir RCT. We note that the SmartAir reduces $PM_{2.5}$ compared to the Normal condition by about 31%, from 8.85 μg/m^3 to 6.13 μg/m^3. This difference is significant with a p-value less than 0.01. SmartAir on average increases furnace fan on time from 24% to 42%. However, its performance in $PM_{2.5}$ is not statistically significantly different from leaving the furnace fan on 100% of the time, 5.71 μg/m^3 vs. 6.13 μg/m^3.

5.4. *$PM_{2.5}$ Distribution*

We note that our measurements indicate a heavy positive tail of the distribution of daily $PM_{2.5}$ in all conditions. We show a boxplot for the values from Deployment I in Fig. 8 and a CDF for Deployment II in Fig. 9, to provide two graphical ways of displaying the data.

Some days are particularly high in $PM_{2.5}$, and the right tail is far from the normal distribution, despite the fact that the daily $PM_{2.5}$ is an average of 2880 one-minute measurements from two PM sensors. Future work should implement statistical tests that are robust to non-Gaussian summary statistics.

As we would expect because SmartAir actuates to reduce $PM_{2.5}$ only when the $PM_{2.5}$ value is high, it does little to change low $PM_{2.5}$ values, and has its main effect by reducing the maximum $PM_{2.5}$ values. Thus the interquartile range is notably smaller for SmartAir compared to Normal.

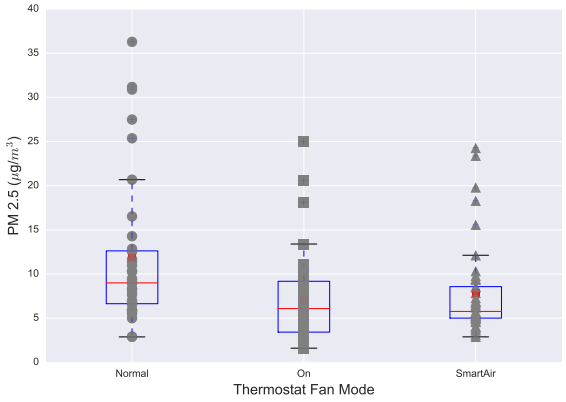


Figure 8: Boxplot of average daily $PM_{2.5}$ for Normal, On, and SmartAir conditions for Depl. I.

5.5. *More Fan Use Without a Air Quality Sensor*

SmartAir combines two things – more use of the furnace fan, and air quality sensing. A legitimate question can be asked: Is the effect we see with SmartAir purely the result of the increased furnace fan utilization, and not particularly related to the use of the air quality sensor to decide when to turn it on?

To address this question, we included in Deployment I two other conditions in the condition list for a period of about two months, a subset of the deployment duration. We call these conditions Runtime 20 and Runtime 40. The Runtime 20 condition sets the thermostat fan to be on for at least 20 minutes every hour. Similarly, Runtime 40 sets the thermostat fan to operate at least 40 minutes out of every hour. SmartAir was set to randomly select between these five conditions, instead of the three conditions originally mentioned in Section 2.3.

The results indicate that the benefit from SmartAir is not simply that the furnace fan is on more often. In fact, Runtime 20 has a higher fan on percentage compared to SmartAir, but the $PM_{2.5}$ reduces by only 19% (Runtime 20 vs. Normal) compared to the $PM_{2.5}$ reducing by 33% (SmartAir vs. Normal). Figure 6 graphs the relationship between average $PM_{2.5}$ vs. Fan On percentage on all the conditions. Although more data could be collected with a variety of "Runtime Y" conditions, where $0 < Y < 60$, our data indicates that the furnace fan must be on a large percentage of the time to achieve performance as good as SmartAir without an air quality sensor.

6. Discussion and Future Work

First, we note that TESS is able to run automated repeated measurement RCTs. In our test of SmartAir, we are able to find statistically significant differences in $PM_{2.5}$ between SmartAir and Normal furnace fan operation. SmartAir is useful in itself because it provides a means to improve one's indoor air with less energy consumption. We note that although we used TESS to test SmartAir, we could readily modify the software to run SmartAir every day by default, unless directed otherwise by the user. We can thus use the same gateway to implement a smart system as we used to test the system.

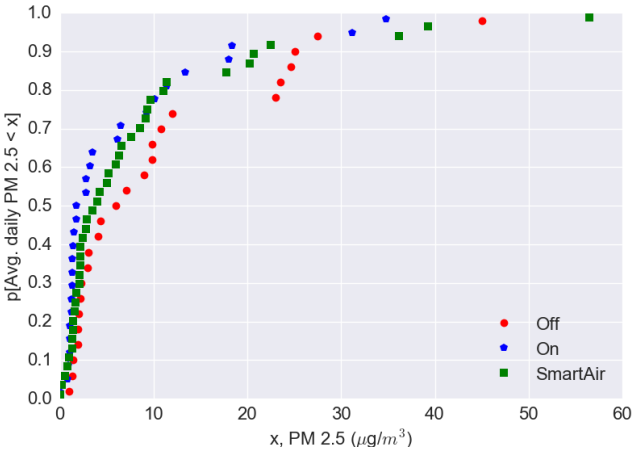


Figure 9: Cumulative distribution function for average daily $PM_{2.5}$ for Normal, On, and SmartAir for Depl. II.

Based on our design and deployment experiences, we point out some important considerations in the broader future use of TESS by individuals to conduct experiments on themselves.

Safety: Automation changes the physical environment, which raises real safety concerns. A system design should include mechanisms to stop/shutdown the actuator system if actuation is dangerous. SmartAir implements a policy to protect residents in case of a house fire as described in Section 2.3. Institutions generally have internal review boards (IRBs) which force researchers to consider all possible safety concerns prior to obtaining approval for a study, including the IRB approval we obtained for this study. Further, although one resident may choose to run a self-science experiment, it may affect other residents, including children. Self-science experiments run by individuals with SmartAir may raise new ethical concerns since the individuals will not have a natural IRB. We suggest that future research should investigate automated ways, given the combination of sensors, actuators, and code, to raise possible safety and ethical concerns to individuals prior to deployment. Further, experiments should include a physical "kill switch" that allows anyone in the home to safely turn off an experiment.

User Preference: The system should not override the users' preferred settings for the automation devices in cases when the automation can affect the user comfort level. For example, home occupants program the preferred heat and cold settings of a smart thermostat. These settings should be honored in the controller algorithm. However, use of a furnace fan does change the temperature and humidity by the act of circulating air from one part of a home to another. Experiments might generically send a periodic survey question to occupants' phones to assess whether their preferences are being met, and look for any change in this as a function of condition.

Effect of Real-Time Monitoring: A home occupant may change their behaviour when we deploy our MDS sensor in their house, just because it reminds them to avoid activities that would worsen their air quality. Further,

the MDS sensor displays the current air quality particle count, which may make participants become more aware of their environment and encourage them to keep their air clean. We note that these effects would be the same regardless of condition, so we can still compare the conditions via the RCT. In general, experiments are required to explore the extent to which occupants change behaviour because of the presence of new IoT sensors and actuators, and as a function of the user interface.

Thing-Enabled Self-Science (TESS): TESS is a general software framework designed to extend individual components and operate experiments. TESS should enable easy setup of a variety of scientific experiments with minimal programming expected from users; Based on the TESS framework, we developed two other IoT applications with off-the-shelf sensors and actuator devices to study individual sleep efficiency depending on light color and indoor temperature. To be widely useful, TESS must be able to communicate with a wide variety of sensors and actuators, with different communication interfaces and protocols. Researchers are typically highly averse to data loss, yet home internet and electrical power experience outages, and residents sometimes unplug devices. System reliability, security, and scalability must also be considered for TESS for future work. For the RCT experimental periods, TESS could evaluate an IoT system faster using a higher number of deployments. Additionally, as the relative impact of an IoT system increases, it will achieve statistical significance over fewer days. TESS could implement a Wald sequential test in the Analysis Manager that would stop the experiment as soon as significance is achieved.

7. Related Work

N-of-1 Trials: The method known as the N-of-1 trial has been used for chronic diseases in an effort to optimize the disease management of the individual, especially when response to a treatment varies. N-of-1 trials randomize use of a drug and placebo over repeated periods of time on the same individual, which provides evidence on the individual's response to treatment. The N-of-1 methodology is an extension of personalized medicine to scientific experiments. This N-of-1 method inspired us to implement a generalized architecture that an individual user can use to decide upon the effectiveness of SmartAir in their own household. Individual responses to an IoT-based intervention can be varied due to real-world complexities [26]. Our results with the four households in which we deployed and performed repeated measrement RCTs showed that SmartAir had different performance in different households as shown in Table 1.

IoT System Testing: There have been many efforts and studies conducted in the WSN and IoT research areas to efficiently test the performance of IoT systems: by developing new physical and virtual testbeds [27, 28, 29], by using a new methodology with dynamic code insertion and emulating IoT resources [30], and by creating architectural innovations designed around efficient system testing [31]. Most of these testing approaches focus on scalability, resource efficiency, and system operation.

Sanchez et al. [28] proposed a new deployment and experimentation architecture for evaluating IoT systems under real-life conditions. This architecture provides a testbed platform and IoT infrastructure to address the holistic IoT experimentation challenges such as large-scale, security, and privacy in various application areas. However, different from observational IoT testing platforms, our TESS architecture focuses on validation with randomized controlled trials, creating evidence-based information and results that are valid even when there are confounding variables.

Improving Indoor Air Quality: Sublett in [6] studied the effectiveness of various air filters and air cleaners for allergic respiratory diseases and suggested using a disposable high-efficiency MERV 12 HVAC filter with well-maintained HVAC system has clinical benefit for asthma. This study supports using the furnace fan with the high-efficiency MERV filter to reduce air pollution for improving residents' health. SmartAir is able to improve indoor air quality by automatically actuating furnace fan when needed, but additionally has the advantage of reducing electrical energy consumption compared to leaving the fan on all of the time.

Simulation Experiments: Crawley in [11] developed EnergyPlus, a popular simulation program in modeling buildings for energy usage and testing. It is organized with a modular structure that allows features and other programs to be easily integrated into the system to model variations in buildings and homes for simulation experiments. Although EnergyPlus enables various simulation experiments, the system is constrained in simulation and does not account for practical variations of individual circumstances and settings for a particular home. In this paper we describe a method to automatically test a smart health system which determines its performance in any particular home, or across many homes to come to a conclusion about the population in general.

8. Conclusion

In this paper, we presented an air quality automation system called SmartAir. We present how it uses air quality sensors to determine when to turn on the furnace fan in a home, and leaves the fan off when the pollution is low to save energy. IoT devices are designed for automation, and we present TESS, a controller that runs a randomized controlled trial across many measurement periods in the home to automatically quantify the performance of SmartAir vs. two other control conditions. By automatically and randomly setting the condition for the independent variable, we can develop scientifically valid conclusions despite the presence of confounding variables. We used TESS to run an RCT for SmartAir in four houses, over a 350-day period. Our results show that SmartAir uses 58% less energy to run the furnace fan compared to leaving the furnace fan on all the time while achieving a very similar reduction in $PM_{2.5}$. SmartAir reduces $PM_{2.5}$ by 31% compared to normal furnace fan operation. This result is statistically significant. Further, we show that the reduction in $PM_{2.5}$ cannot be achieved by simply using the fan for a higher percentage of the time without air quality sensing.

9. Acknowledgment

Research reported in this publication was supported by the NIBIB / NIH under award #1U54EB021973-01.

References

[1] W. Kanchongkittiphon, M. J. Mendell, J. M. Gaffin, G. Wang, W. Phipatanakul, Indoor environmental exposures and exacerbation of asthma: an update to the 2000 review by the Institute of Medicine, Environmental health perspectives 123 (1) (2015) 6.
[2] C. A. Pope III, M. Ezzati, D. W. Dockery, Fine-particulate air pollution and life expectancy in the United States, New England Journal of Medicine 2009 (360) (2009) 376–386.
[3] M. S. Andersen, Co-benefits of climate mitigation: Counting statistical lives or life-years?, Ecological Indicators 79 (2017) 11–18.
[4] A. P. Jones, Indoor air quality and health, Atmospheric environment 33 (28) (1999) 4535–4564.
[5] A. Clerici, G. Alimonti, World energy resources, in: EPJ Web of Conferences, Vol. 98, EDP Sciences, 2015, p. 01001.
[6] J. L. Sublett, Effectiveness of air filters and air cleaners in allergic respiratory diseases: a review of the recent literature, Current Allergy and Asthma Reports 11 (5) (2011) 395–402.
[7] J. W. Thornburg, C. E. Rodes, P. A. Lawless, C. D. Stevens, R. W. Williams, A pilot study of the influence of residential HAC duty cycle on indoor air quality, Atmospheric Environment 38 (11) (2004) 1567–1577.
[8] A. Hernandez, Withings Aura review: One of the best smart alarm clock experiences (2016).
URL https://techaeris.com/2016/10/15/withings-aura-review-one-best-smart-alarm-clock-experiences
[9] A. Beltran, V. L. Erickson, A. E. Cerpa, Thermosense: Occupancy thermal based sensing for HVAC control, in: Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings, ACM, 2013, pp. 1–8.
[10] A. Rabbani, S. Keshav, The SPOT* personal thermal comfort system, in: Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments, BuildSys '16, 2016, pp. 75–84.
[11] D. B. Crawley, L. K. Lawrie, F. C. Winkelmann, W. F. Buhl, Y. J. Huang, C. O. Pedersen, R. K. Strand, R. J. Liesen, D. E. Fisher, M. J. Witte, et al., Energyplus: creating a new-generation building energy simulation program, Energy and Buildings 33 (4) (2001) 319–331.
[12] A. L. Northcross, R. J. Edwards, M. A. Johnson, Z.-M. Wang, K. Zhu, T. Allen, K. R. Smith, A low-cost particle counter as a realtime fine-particle mass monitor, Environmental Science: Processes & Impacts 15 (2) (2013) 433–439.
[13] S. Steinle, S. Reis, C. E. Sabel, S. Semple, M. M. Twigg, C. F. Braban, S. R. Leeson, M. R. Heal, D. Harrison, C. Lin, et al., Personal exposure monitoring of PM 2.5 in indoor and outdoor microenvironments, Science of the Total Environment 508 (2015) 383–394.
[14] K. T. Min, P. Lundrigan, N. Patwari, IASA-indoor air quality sensing and automation: demo abstract, in: Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks, ACM, 2017, pp. 277–278.
[15] Code for Utah-modified Dylos sensor (2018).
URL https://github.com/VDL-PRISM/T
[16] Grimm technologies (2017).
URL http://www.dustmonitor.com/IAQ/
[17] Impact sampler, for PM2.5 sampling at 10l/min (2017).
URL http://www.skcinc.com/catalog/product_info.php?products_id=397
[18] P. Lundrigan, K. T. Min, N. Patwari, S. K. Kasera, K. Kelly, J. Moore, M. Meyer, S. C. Collingwood, F. Nkoy, B. Stone, K. Sward, Epifi: An in-home sensor network architecture for epidemiological studies, Tech. Rep. arXiv:1709.02233 [cs.NI], arXiv.org (Sep. 2017).
[19] Home assistant (2017).
URL https://home-assistant.io/
[20] Open-source home automation platform running on Python 3 (2017).
URL https://github.com/home-assistant/home-assistant
[21] InfluxDB (2017).
URL https://www.influxdata.com/

[22] Grafana (2017).
URL https://grafana.com/
[23] N. A. F. A. T. Committee, Understanding MERV NAFA Users Guide for ANSI/ASHRAE Standard 52.2-2012 Method of Testing General Ventilation Air-Cleaning Devices for Removal Efficiency by Particle Size (Nov. 2014).
URL https://www.nafahq.org/understanding-merv/
[24] P. N. Breysse, T. J. Buckley, D. Williams, C. M. Beck, S.-J. Jo, B. Merriman, S. Kanchanaraksa, L. J. Swartz, K. A. Callahan, A. M. Butz, et al., Indoor exposures to air pollutants and allergens in the homes of asthmatic children in inner-city baltimore, Environmental Research 98 (2) (2005) 167–176.
[25] NAAQS table (2012).
URL https://www.epa.gov/criteria-air-pollutants/naaqs-table
[26] P. A. Scuffham, J. Nikles, G. K. Mitchell, M. J. Yelland, N. Vine, C. J. Poulos, P. I. Pillans, G. Bashford, C. Del Mar, P. J. Schluter, et al., Using n-of-1 trials to improve patient management and save costs, Journal of General Internal Medicine 25 (9) (2010) 906–913.
[27] M. Nati, A. Gluhak, H. Abangar, W. Headley, Smartcampus: A user-centric testbed for internet of things experimentation, in: Wireless Personal Multimedia Communications (WPMC), 2013 16th International Symposium on, IEEE, 2013, pp. 1–6.
[28] L. Sanchez, L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis, et al., Smartsantander: IoT experimentation over a smart city testbed, Computer Networks 61 (2014) 217–238.
[29] S. Latre, P. Leroux, T. Coenen, B. Braem, P. Ballon, P. Demeester, City of things: An integrated and multi-technology testbed for IoT smart city experiments, in: Smart Cities Conference (ISC2), 2016 IEEE International, IEEE, 2016, pp. 1–8.
[30] E. S. Reetz, D. Kuemper, K. Moessner, R. Tönjes, How to test IoT-based services before deploying them into real world, in: Wireless Conference (EW), Proceedings of the 2013 19th European, VDE, 2013, pp. 1–6.
[31] P. Rosenkranz, M. Wählisch, E. Baccelli, L. Ortmann, A distributed test system architecture for open-source IoT software, in: Proceedings of the 2015 Workshop on IoT challenges in Mobile and Industrial Systems, ACM, 2015, pp. 43–48.