ECE 3375: Project Progress Report

Arnav Goyal - 251244778
Chinmaya Ranebennur - 251170914
Graeme Watt - 251201502
Edward Golshenas-Rad - 251149374

March 24, 2024

Problem Definition

In a smart-home setting, the current tech landscape has led to a situation where many smart-devices operate independently of one another, lacking a cohesive system (a central hub) for management. This project proposes the development of a *Smart Home System* designed to seamlessly integrate a suite of sensors and actuators that allow for automatic adjustments to the home environment based on real time data.

The primary challenge we have identified is the lack of centralization for separate smart-home products. A truly intelligent (smart) home setup would have a centralized hub that can manage and coordinate the functionalities of various other smart devices, thereby minimizing the need for human input. More specifically our system would have the below features:

1. Automatic Lighting Control
2. Ambient Temperature Control
3. Humidity Control
4. Home Security

Each of these features will be touched upon briefly within the **Functional Description** section. We think that creating a centralized hub for smart home management will benefit both the user and the manufacturer. The existence of a smart home hub allows for easier setup and tweaking of settings for the homeowner. Once ubiquitous, the existence of a centralized hub would also require some industry standard protocols to be developed making products easier to create.

Functional Description

Our system will basically feature some sort of control mechanism to optimize the home environment. We will be implementing a fairly simple control algorithm which involves some sort of input sensor reading, and some sort of actuator that changes the sensor reading by altering the home environment in some way.

Essentially we will be reading inputs and writing outputs to/from the GPIO or I2C. The user will first define the threshold values, such as desired resting temperature, humidity, lighting timeout, etc. And our software will check to see if we have met those conditions. For example, if our temperature sensor shows that we are below the desired temperature, we would turn on the heater. The reading of the sensor and output to the heater are both done through the GPIO. Each section is described in more detail below.

Automatic Lighting Control

Utilizing daylight sensors, the system will gauge the level of natural light outside the house. During daylight time, the house will have sufficient natural lighting and the system will keep the lights turned off to conserve energy. At night (or when natural lighting conditions are deemed insufficient) the system will turn the lights on, but only in the presence of individuals. If there is nobody for a certain amount of time, the lights will be shut back off to conserve power.

Ambient Temperature Control

The system uses a temperature sensor to continuously monitor the ambient temperature. If the temperature deviates from a user-defined range, the system automatically activates a heater to adjust the temperature back within the desired range, This approach ensures the living space remains comfortable.

Humidity Control

In conjunction with temperature control, the system includes a humidity sensor to assess the moisture levels in the air. If the humidity falls outside a user-defined range, the system activates a humidifier to restore the humidity to a more comfortable level. This feature is particularly beneficial for maintaining indoor air quality, and occupant comfort.

Home Security

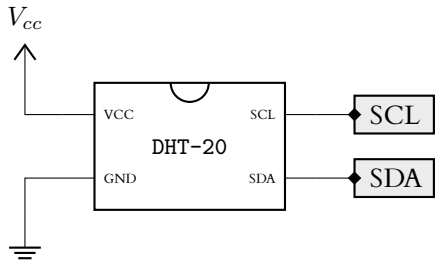
Through the use of hall sensors, the system can gauge the state of a door (open or closed). If the home security is in a 'locked' state and the door opens, we can activate an alarm. However if the home security is in a 'relaxed' state, changes to the door will not set off the alarm. This is a simple solution to home security, but it can be quite effective.

Input & Output Requirements

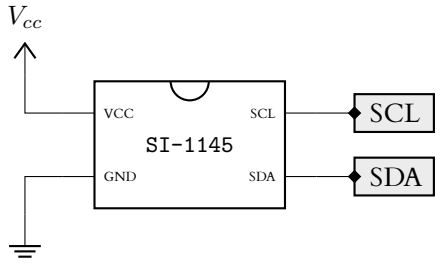
The components we will need to make this project are listed below. Furthermore schematics will also be provided on how to interface them with the micro-controller. In the table below I2C is the inter-integrated circuit protcol, and digital input/output means we will be using the GPIO

Subsection	Component	Part Name & Clickable Link	Protocol
Lighting	Sensor	Daylight Sensor	I2C
	Sensor	Proximity Sensor	Digital Input
	Actuator	Lights (an LED)	Digital Output
Temperature	Sensor	Temperature Sensor	I2C
	Actuator	Heater (DC Relay)	Digital Output
Humidity	Sensor	Humidity Sensor	I2C
	Actuator	Humidifier (DC Relay)	Digital Output
Home Security	Sensor	Hall Effect Sensor	Digital Input
	Sensor	Toggle Switch	Digital Input
	Actuator	Alarm (Active Buzzer)	Digital Output

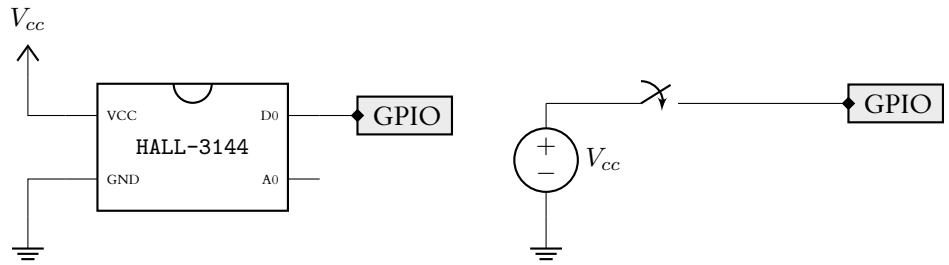
In total we have 2 I2C devices (one device reads both temperature and humidity), and we need at least 7 GPIO (3 Digital Input + 4 Digital Output) pins. We will also need a toggle switch to set the status of the security system as armed or disarmed, but that is a trivial switch reading (most microcontrollers will have an onboard switch, thus this is not listed in the components section at the moment). **Circuit 1** to **Circuit 8** show how we can interface each of these components with a GPIO or I2C interface for a microcontroller.



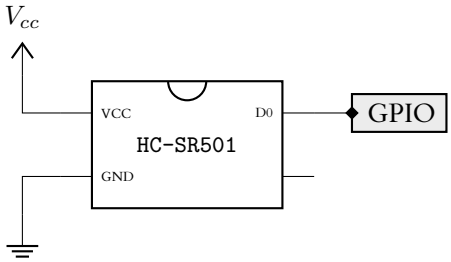
Circuit 1: Temperature & Humidity Sensor Interfacing



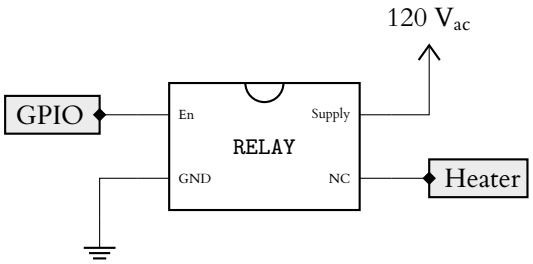
Circuit 2: Daylight Sensor Interfacing



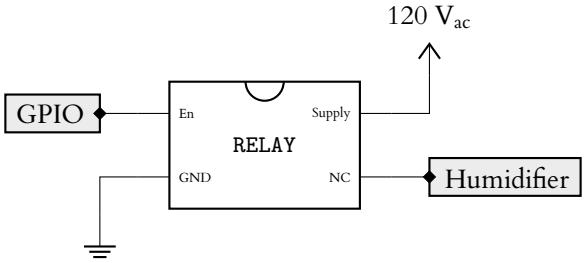
Circuit 3: Hall Sensor & Toggle Switch Interfacing



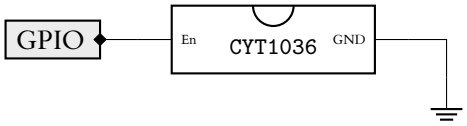
Circuit 4: Proximity (PIR) Sensor Interfacing



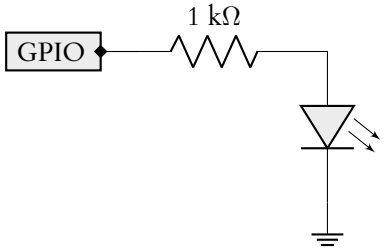
Circuit 5: Relay Interface for Heater



Circuit 6: Relay Interface for Humidifier



Circuit 7: Interfacing for Alarm (Active Buzzer)



Circuit 8: Interfacing for Lights (LED)

Initial Software Design

Our software will have a couple tasks to perform during its use.
Initially (On startup) we will have to:

1. Set up I2C
2. Set up GPIO

After initialization has finished we need do the following continuously:

1. Read Sensor Values through I2C
2. Read Sensor Values through GPIO Inputs
3. Process the readings, determine if we are within defined ranges
4. Change the status of each actuator if needed (to active or idle)
5. Wait for some time before reading values again.

Our inputs (sensor values) are sampled either through I2C protocol or Digital Inputs from the GPIO, thus we will have to set these two things up in the initialization part of our C program. We use these inputs to compute the outputs (state of the actuators) and then actually change the state of the actuators based on conditional processing of the input data. This conditional processing is shown via the (very bare-bones, and python-like) pseudo-code in **Code Snippet 1**.

Our software spends most of its time in the infinite-loop, reading sensors, determining actuator states, and waiting. It does run forever, and it is intended to run forever (or while turned on), its meant to be run until the user stops it.

Furthermore, our software features a pretty simple interaction between inputs, outputs and states which can be modeled as a *Finite-State-Machine (FSM)* but this will not be drawn in LaTeX right now. Our software essentially uses inputs (sensor values) to determine a current state for all the actuators in the system, and each state features a unique output for each actuator. This is basically the interaction of an FSM, thus it can be modeled as one.

```
# initial values for global sensor readings
temp = 0
humiditity = 0
armed = False
hall = True # assume True means door is closed
proximity = False # assume False means nobody is detected
daylight = 0

def main():
  # initialization
  initGPIO()
  initIIC()

  # inf loop
  while (1):
    readSensors()

    if (temp <= MIN_TEMP):
      activateHeater()
    else:
      deactivateHeater()

    if (daylight <= MIN_LIGHT && proximity == True):
      activateLight()
    else:
      deactivateLight()

    if (humidity <= MIN_HUMIDITY):
      activateHumidifer()
    else:
      deactivateHumidifer()

    if (armed == True && Hall == False):
      activateBuzzer()
    else:
      deactivateBuzzer()

    # wait for a while before
    # redoing the readings & state-determination
    delay(DELAY_LENGTH)
```

Code Snippet 1: Initial Implementation Pseudo-Code

Prototyping Plan

Obviously we are not going to buy each component and actually test this project, We can fake the appearance of hardware well enough to functionally verify our design though. In general this will be done in two (pretty large) steps, where we will have to verify:

1. The Acquisition of Data
    (a) Test the GPIO Connectivity
    (b) Test the I2C Connectivity
2. The Logical/Conditional Processing of Data
    (a) Test the thresholding (input) logic
    (b) Test the actuator toggle (state transition) logic
    (c) Test the output logic

To verify the first part, We can use a general I2C sensor that someone already has (from an earlier class where we were forced to buy them) to test whether our I2C has been set up correctly. Obviously once a single device has been set up correctly it is safe to assume that every other device would also be set up correctly as we would basically just have to change the address we read from. For GPIO We can build a simple switch circuit (as shown in **Circuit 3**) and test whether we can read the state of the toggle switch.

To verify the second part, we can use the ability to model the program as an FSM. Due to this, our design is inherently testable with the built-in peripherals on most dev-boards. We are probably going to implement this prototype with another microcontroller (such as the STM32F411RE), but in the worst-case it should be testable on the DE-10 Standard. To test, we can use LED lights to represent the state of each actuator, and use switches to represent if the input from each sensor violates (or doesn't violate) the threshold set by the user. We can then use these peripherals to test the logical and conditional processing of our program.

Altogether, once we have verified both steps above, we can be fairly certain that our design has worked! There might be things to consider in the future that we haven't considered/encountered yet, but those factors will be noted and added to the Final project report.