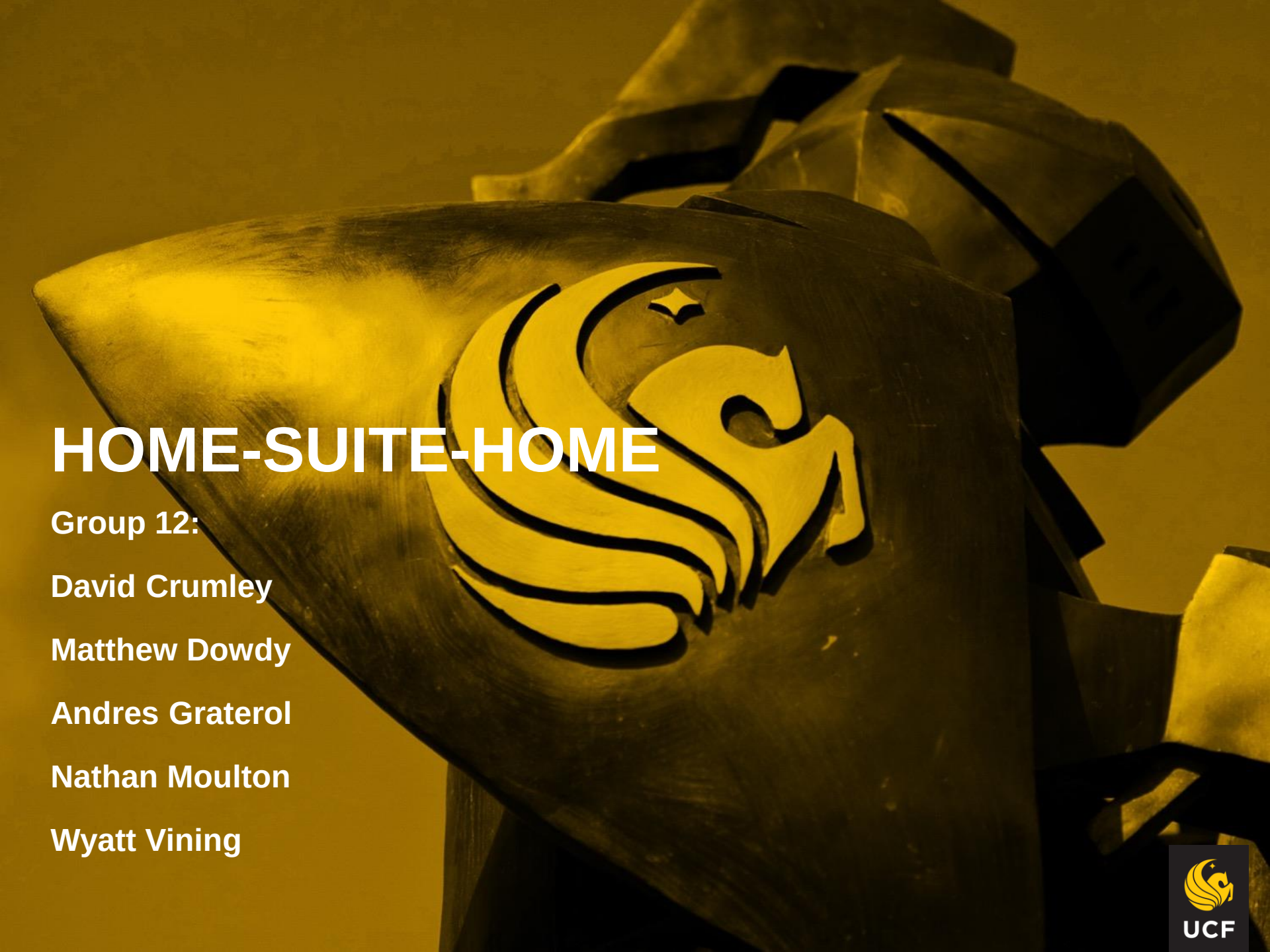

HOME-SUITE-HOME

Group 12:

David Crumley

Matthew Dowdy

Andres Graterol

Nathan Moulton

Wyatt Vining

UCF

Purpose & Motivation

Home-Suite-Home was designed with the intention of providing a method for property owners to remotely view the status of their properties. The properties would be installed with multiple sensors that could be easily accessed using a User-Interface and email system. With Home-Suite-Home installed into the homeowner’s properties, the customer would never have to worry about what is happening while they are away!

UCF

System Alternatives

Our Competitors:

- Google Nest Hub
  - This system provides an interconnection of smart devices around the home. Google Hub has compatibility with devices such as security cameras, smart lights, and speakers.
- Samsung SmartThings Hub
  - This system provides an interconnection of smart devices around the home. It comes with a companion mobile application and allows for the addition of other Samsung brand smart devices such as motion sensors and water leak detectors.

Us:

Since our system is open-source, it eliminates the need for such high installation and maintenance costs associated with the competition. There is no subscription necessary, and users have the ability to configure and add their own sensors. Unlike our competitors, Home-Suite-Home allows the user to add sensor of any type. In comparison to the other systems, our sensor suite is focused on providing the user with information solely on property health. In addition to this, we have real-time analytics so that the customer can view a full history of all their desired sensors. The ability to add sensors on the fly, view real-time data, and receive real-time alerts is what makes Home-Suite-Home rise above the competition.

UCF

Core Features

This system boasts:

- An email system that can make requests to the database running on a raspberry pi and receive responses in return. The email also receives alerts when any sensor is out of range of the user-defined tolerance.
- Analytics that show the history of all the sensors either through email commands or through button presses on the user-interface.
- A user-interface that contains:
  - A settings page to enter the raspberry pi credentials, user emails, silence alerts, and enter desired data refresh rate and notification rate.
  - A home screen that allows users to add, edit, and delete sensors. Each card also allows the user to view the analytics for the specific sensor.

UCF

Technologies

- Git and Github for version control.
- Raspberry Pi 4V+ to aggregate all of the sensor data and host the database.
- MongoDB to host the database server.
- Plotly to display the analytics graphs.
- Dash to assist in the creation of the user-interface.
- VNC Viewer
- Languages used: Python, C++, Bash Script, HTML, CSS
- Arduino: NodeMCU V12E for each sensor
- Sensors (written in Arduino C++):
  - Temperature
  - Humidity
  - Leak detection
  - Light
  - Pressure
  - Range

UCF

Algorithms

- For handling email commands:
  - Every 30 seconds the script gets all user emails from the database.
    - For each user email check the system mailbox for a message received by the user.
    - If a message has been received, create a response based on the contents of the email. Send the response to the user.
- For Data Analytics:
  - In the case of a single graph:
    - Get data from database based on the requested sensor’s parameters.
    - If no data is returned: return a None object.
    - Convert the received data based on the user’s specified unit preference.
    - Iterate over all the data , removing NaN values and calculating a cumulative average.
    - If only NaN data is found: return a None object.
    - Obtain the max and min from the resulting data set.
    - Create a plotly graph object:
      - Add each trace (data, avg, min, max) to the graph object.
      - Add titles and return to the graph object.

UCF

Algorithms Continued

- For populating the User Interface:
  - The callback: create_new_card:
    - The script gets the state of the sensor parameters (name, type, ip, port, etc…).
    - The callback is activated when the create button is pressed.
      - Upon button press, a new config entry in the database is created using the sensor's entered fields.
      - Sends a message to set_card_container to update the screen.
  - set_card_container:
    - Reads all data from the config collection in the database.
    - Outputs sensor cards to the screen corresponding to the config entries.
    - For populating the User Interface:
- For connecting with the sensors:
  - A sensor object is instantiated with a URL, port and plug corresponding to where the sensor can be found on the network.
  - Address for the sensor is assembled in the format: http://IP_ADDRESS:PORT/plug.
  - Upon requesting sensor value:
    - If the sensor is unavailable NaN is returned in place of a sensor value.
    - Otherwise, sensor value is sent to the requested location.

UCF

Algorithms Continued

- For generating and sending alerts:
  - The alert object is instantiated with a copy of the sensor's config data and its current value.
  - The database is checked to see if an alert was previously logged within the rate limit.
    - If no log exists, the script generates an email to send.
    - The script loops the users registered in the database collection and sends an alert to each of them.
    - The script saves a log to the database to reset the rate limit.
- For settings:
  - This script uses python's ConfigParser to parse INI files.
  - The INI files will contain the necessary data for sensor polling rate and notification rate limit.
  - If the file cannot be read, -1 is returned.
- For running the sensors:
  - Script begins by retrieving the set sensor poll rate and whether or not the user has chosen to silence alerts from the settings.config file.
  - If not poll rate has been specified. The script settles for the default value of 60 seconds.
  - A while loop runs while the system is active.
    - For each iteration, the existing sensor config data is pulled from the database. The retrieved sensor value is compared against the corresponding min and max range if alerts are turned on.
    - Alerts are sent if necessary.
    - New sensor values are changed to the database.
    - While loop waits the sensor poll rate time before running once again.

UCF

Algorithms Continued

- For the sensor hardware template:
  - A setup function will run as soon as the Arduino with the sensor is powered on.
  - The device attempts to connect to the user's network.
  - The sensor is initialized using functions from the sensor manufacturer's library.
  - The URL plug is specified – this is how the sensor data will actually be retrieved.
  - Script generates a response to a sensor value request.
  - If no URL plug is specified, the root page is reached, and an error message is passed to the client.
- For the Database:
  - The script receives the IP and port of the mongodb server process and establishes a connection from which functions can be carried out.
  - The script uses the functions found in the pymongo library to make the necessary database queries.
  - If a database collection is attempted to be read from, but none exists, then the collection is created within the script.

UCF

Future Plans

- We would hope to split the user interface into different categories of sensors.
- We would like to expand the list of default supported sensors.
- We would like to add quality-of-life changes to the User-Interface.
- We want to add more notifications that get sent to the user – such as alerting the user when a sensor goes offline and needs to be serviced.

UCF

Thank You for Choosing Home-Suite-Home!

UCF