Software Formalization

Year: 2023 Semester: Fall Team: 8 Project: Smart Seat Occupancy Sensor
Creation Date: 9/30/2023 **Last Modified:** September 30, 2023
Author: Giang Nguyen **Email: nguye683@purdue.edu**

Assignment Evaluation: See Rubric on Brightspace Assignment

1.0 Utilization of Third Party Software

Software/Firmware Name	Description	Use Case	Licensing Data	Steps to Use
HTTP Protocol [1]	High-level protocol for web communication	Facilitate data transfer between Control Units and Web Server	N/A	N/A
JSON Format [2]	Lightweight, language-independent data format	Packaging data from Control Units to Web Server	N/A	N/A
SQL Databases [3]	Database management system and SQL language	Create and manage the occupancy table on the Web Server	Depends on the specific SQL DB used	1. Install a compatible SQL database, create tables, define schema, and establish connections.
Visual Studio Code [4]	Integrated development environment (IDE)	Write, edit, and debug code for the Control Unit software	Free and open-source (MIT License)	1. Download and install Visual Studio Code. 2. Install relevant extensions for your programming language (e.g., C/C++ extension). 3. Create or open your project in VS Code.

Swift Programming Language [5]	General-purpose, multi-paradigm language	Develop the Phone App for user interaction with the system	Open-source (Apache License 2.0)	1. Install Swift on your development machine. 2. Use Xcode or a compatible IDE to write and test Swift code for the app.
IFTTT (If This Then That) [6]	Automation and integration platform	Automate tasks or integrate with other services based on specific events or triggers	Freemium model with free and paid plans	1. Sign up for an IFTTT account. 2. Create applets (automations) using the IFTTT web interface or mobile app. 3. Connect your Control Unit or other services/devices to IFTTT.

2.0 Description of Software Components

Software Calculations - Once the data is read from the microcontroller, there are multiple calculations that would need to be done.

- Force Sensitive Resistor ADC Protocol - Will take an analog reading and convert it to a digital signal so that it can be processed by the microcontroller. Since our system incorporates 2 FSRs, there are multiple thresholds we need to consider. Based on prototyping, we will decide what threshold numbers is considered "occupied". Since both the FSRs may have slightly different factory resistance values, there are some slight adjustments between them as well. When working with the ADC, we first configure the channel and necessary pins.
  Once the ADC is configured, the software proceeds to perform analog-to-digital conversions for each FSR:
  - Sampling: The ADC samples the analog voltage at regular intervals, controlled by a clock signal, and the sampling frequency can be configured
  - Conversion: The sampled analog voltage is converted into a digital value by the ADC. The result is a binary number that represents the voltage level. This digital value is typically proportional to the applied force on the FSR.

- Human Presence Radar Sensor UART - Similar to the ADC protocol, the sensor will provide an distance measurement. The software calculation will include what decided distance is necessary for seat occupation. Once communication is established, the component sends commands or queries to the sensor to retrieve distance measurements. The sensor responds with the current distance between itself and any detected object, which could be a person approaching or occupying the seat. you done w this? The ESP32 has multiple UART peripherals, each with its own set of pins for communication (usually labeled as TX and RX for transmit and receive).
    - Baud Rate and Configuration: The ESP32 UART peripheral needs to be configured with specific parameters such as baud rate, data bits, stop bits, and parity. These parameters must match the settings of the radar sensor to ensure proper communication. The component is responsible for setting up and initializing UART with the correct configuration.
      Sending Commands to the Radar Sensor:
      Once UART is configured, the component sends commands or queries to the radar sensor. These commands are typically strings or bytes that instruct the sensor to perform certain actions, such as requesting distance measurements.
    - Receiving Data: The ESP32 listens for incoming data on its UART receive pin (RX). When the radar sensor responds, the UART peripheral captures the incoming bytes.
    - Data Parsing: The component parses the received data to extract the distance measurement information. This might involve converting bytes into numerical values and applying any necessary data formatting.

- Thermal Camera I2C - The thermal camera will provide an 8x8 reading of temperature readings in *C as floats. The micro software will have to include an averaging function to both aggregate and summarize the data. A threshold is necessary to determine the detection of a person. We expect the average temperature to rise from 20-23 *C (room temp) to 25-30 *C (body temp). Obviously the threshold will be determined through testing.
    - The I2C connection is established via a handshake to the address of the Sensor on the I2C bus.
    - A timer will then periodically poll the temperature readings. When the timer indicates it is time to poll, the micro will send a request to the Sensor, and receive a response.
    - The processing steps outlined above will then occur.
    - There are third-party I2C libraries that make the bus communication relatively easy, and we will likely use them.

3.0 Testing Plan

Component	Tentative Priority Level	Testing Criteria
Force Sensitive Resistor ADC	(Critical)	
Threshold Calibration Test		Determine suitable occupancy thresholds based on FSR readings
ADC Configuration Test		Validate ADC channel and pin configuration
Analog-to-Digital Conversion Test		Confirm accuracy of FSR analog-to-digital conversion
Human Presence Radar Sensor UART	(High)	
UART Configuration Test		Ensure proper UART configuration for radar sensor
Command Sending Test		Verify microcontroller's ability to send and receive commands
Data Reception and Parsing Test		Validate data reception and parsing from the radar sensor
Thermal Camera I2C	(Medium)	
I2C Handshake and Addressing Test		Confirm successful I2C connection and addressing with camera

Temperature Reading Averaging Test		Verify aggregation and averaging of temperature readings
Threshold Detection Test		Determine accuracy of person detection based on thresholds
I2C Communication Reliability Test		Ensure reliable I2C communication with the thermal camera

4.0 Sources Cited:

[1] Ifttt, “Documentation - build your integration,” IFTTT, https://ifttt.com/docs (accessed Sep. 30, 2023).

[2] “This page requires JavaScript.,” Swift.org, https://docs.swift.org/swift-book/documentation/the-swift-programming-language/thebasics/ (accessed Sep. 30, 2023).

[3] Microsoft, “Documentation for visual studio code,” RSS, https://code.visualstudio.com/Docs (accessed Sep. 30, 2023).

[4] “Essential SQL for the beginners,” SQL Tutorial, https://www.sqltutorial.org/ (accessed Sep. 30, 2023).

[5] “What is JSON?,” What is JSON, https://www.w3schools.com/whatis/whatis_json.asp (accessed Sep. 30, 2023).

[6] MozDevNet, “An overview of HTTP - http: MDN,” HTTP | MDN, https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview (accessed Sep. 30, 2023).

Appendix 1: Software Component Diagram

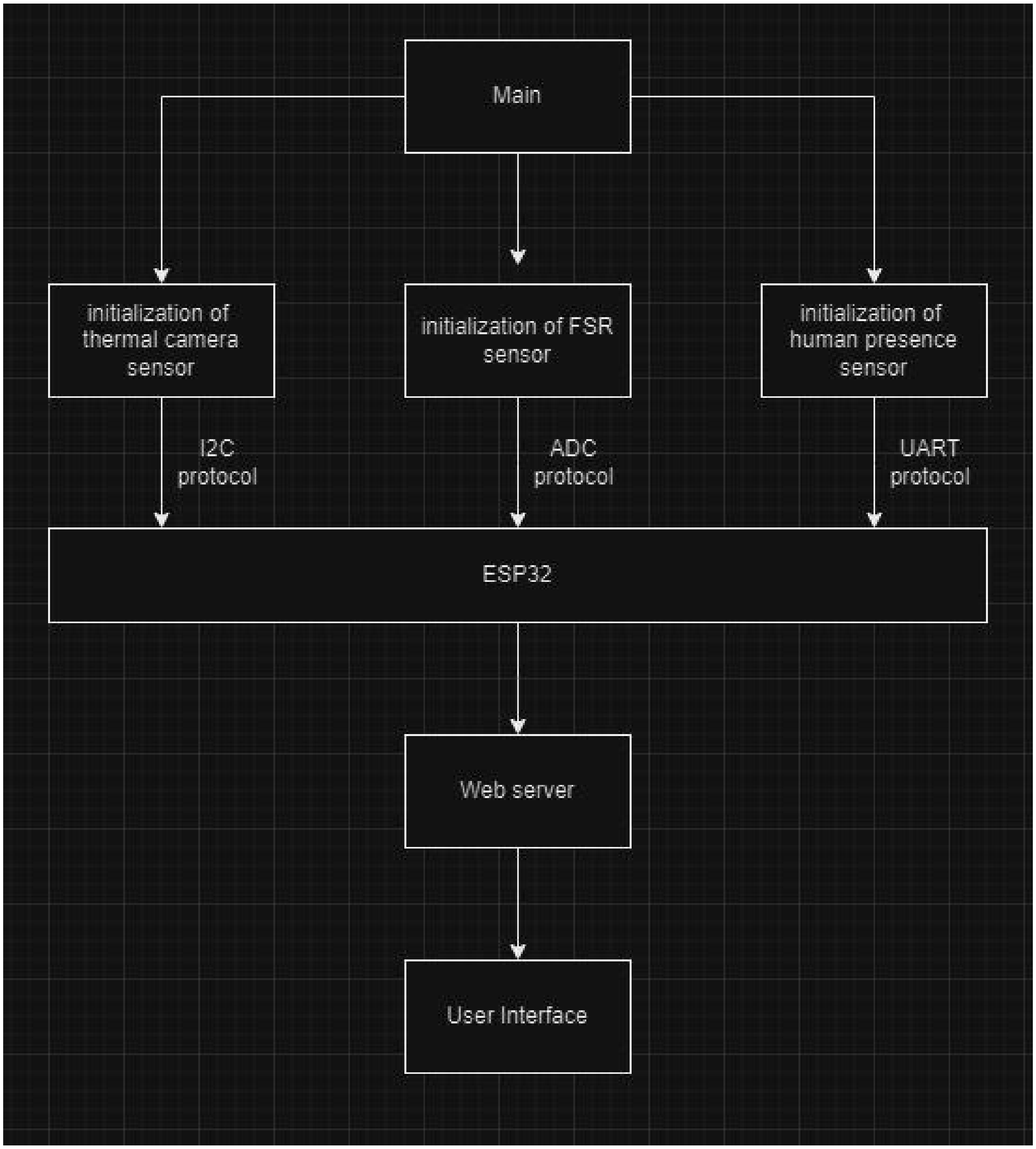
Main
initialization of thermal camera sensor
initialization of FSR sensor
initialization of human presence sensor
I2C protocol
ADC protocol
UART protocol
ESP32
Web server
User Interface

Fig1: Overall design

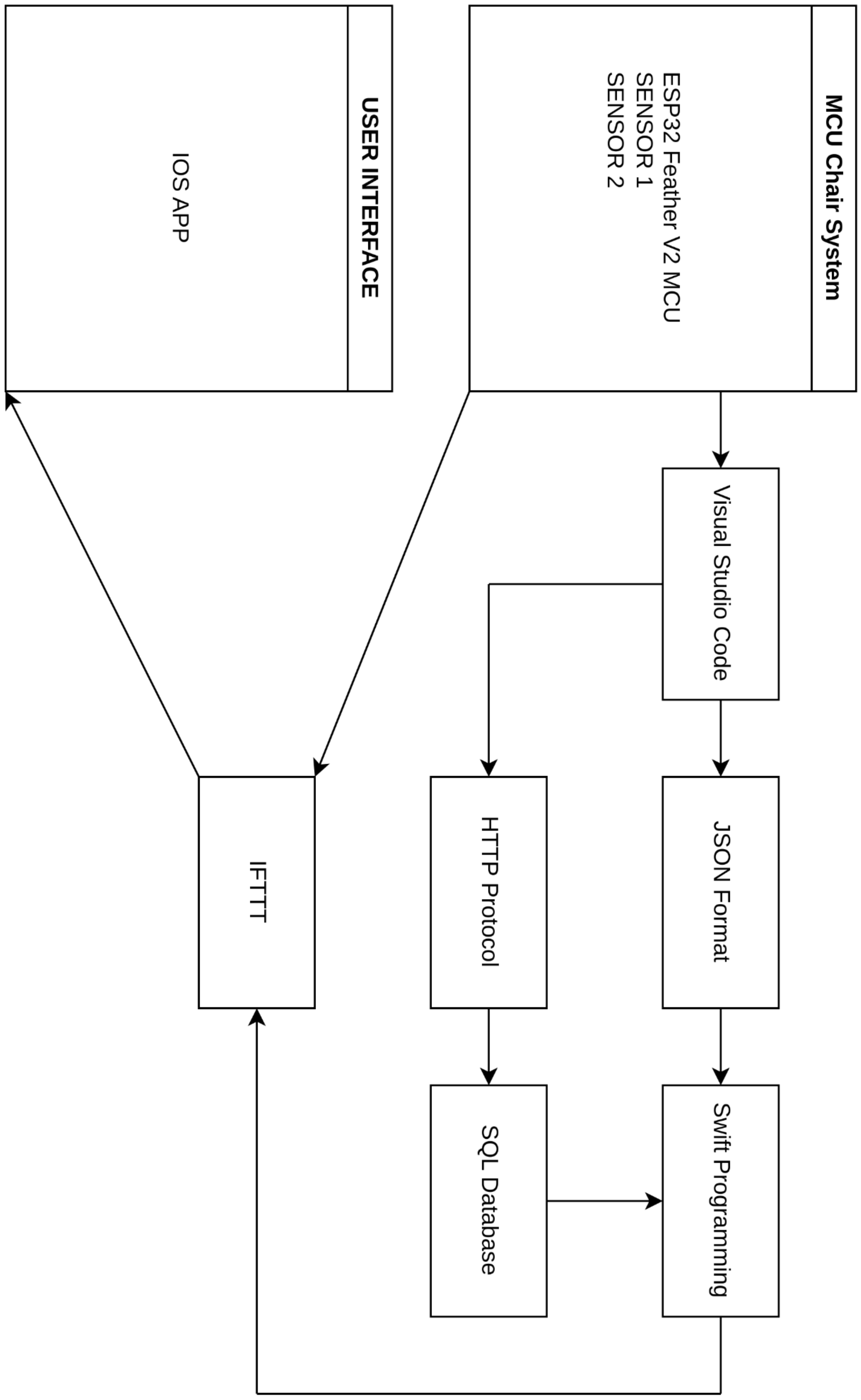

Fig2: Software design details