EECS 159A Senior Design

IOT Smart Charger Project Plan

By Circuit Banditos

12/11/2017

Brandon Metcalf, Electrical Engineering
Andy Begey, Electrical Engineering
Luis Contreras, Computer Engineering
Shermaine Dayot, Electrical Engineering

Mentor: Michael Klopfer/G.P. Lee

Executive Summary:

The EVSE smart charger project is an energy saving application aimed at redesigning the internal workings of current EVSE chargers by adding limited edge intelligence to the controller and providing a platform for which to build a controller from microprocessor units never used in such a system before. This group plans to design and implement a fully self-contained electric vehicle charge controller - electric vehicle supply equipment (EVSE), capable of interfacing with any electric car that understands J1772 communication protocol. It will also include wireless internet capability and current sensing hardware to estimate charge states and levels for charge scheduling. Execution of this project involves careful study of current Open EVSE circuit technology and improvement on its implementation. The end goal is to deliver a fully functional isolated charging unit that could replace any commercial EVSE charger sold on the market today, with the added ability to monitor power consumption of the vehicle and schedule charge times wirelessly.

Table of Contents

Chapter 1: Introduction

Since the recent increase in demand for electric cars over the last 10 years, the infrastructure surrounding transportation has had to make some modifications. One of these modifications is the invention of a charging “aid” to regulate power drawn from the domestic power grid so as to not over-draw power and damage it. Unfortunately, there are long term issues that have not yet been addressed. The increase of electric vehicle (EV) plug loads on the domestic power grid coupled with commuting habits can lead to a dangerous amount of power drawing from the electric grid (**technologyreview**). Car chargers have power requirements comparable to the aggregate requirements of all other household appliances combined.

Another consideration is the time of day designated to charging. Electric companies charge their customers on a kilowatt hour basis. The rates can fluctuate during the time of day. Typically, peak hours are during the daytime which can be financially catastrophic to any electric car owner seeking to charge their vehicle during that period of time. Recharge times can vary from 4 - 8 hours depending on the model of the vehicle. (fueleconomy)

To both protect the electric grid from excessive power drawing and protect EV owners from excessive charging rates, we can build a smart charger capable of scheduling charging times and allow users to easily monitor and control the power consumption of their vehicles using a mobile app.

1.1 Motivation

The current infrastructure for charging electric vehicles is not as efficient as it can be. The fact that electric vehicle supply equipment/systems (EVSE) chargers do not factor these problems into their design is problematic and certainly a concern worth addressing. One of the most powerful applications of many IOT projects is their ability to save users money and time. This group is excited to be a part of seeing these design modifications come to fruition. Our aim is to help alleviate these shortcomings by producing a charging system that can dynamically respond to fluctuations in the power grid, minimizing power consumption, and cost of power usage.

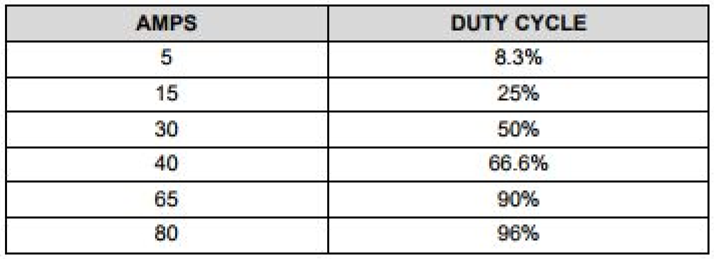
1.2 Contributions

The work of this project, technically, falls under the field of embedded systems engineering, and applicably under the field of IOT. Each team member is interested in gaining more experience in embedded systems work towards their engineering careers. IOT is a fast expanding field, due to the large influx of microcontroller-controlled sensors over the last 10 years. While these sensors have heavily impacted domestic communities, their usefulness for prototyping can not be understated. This project will help each team member get experience in embedded systems engineering for both hardware and software. Several companies are interested in our project, since we will be using a Microsemi FPGA and Smartenit back-end software. This project aims to modify an existing open source EV controller called OpenEVSE while adding new components for current sensing and IoT connectivity. If implemented correctly, this project may be included as application notes on the Microsemi and Smartenit websites. Because of this, our group is very excited about the potential contributions this project will make towards our careers.

Chapter 2. Background and Related Work

2.1 Background

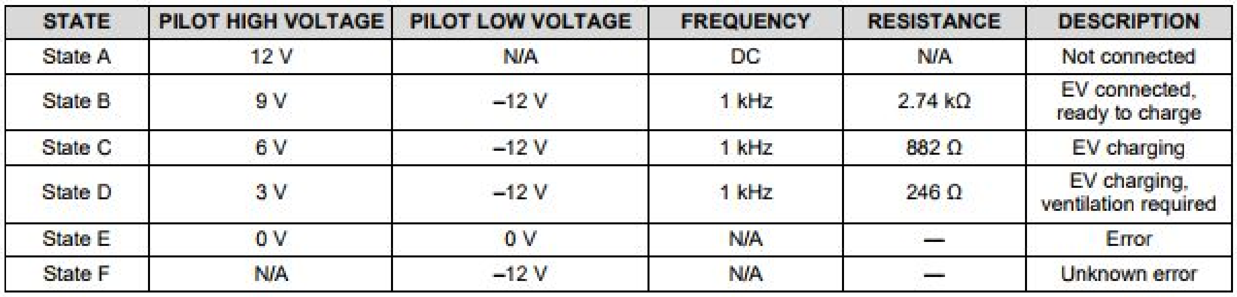
EVSE standard protocol J1772 is the communication protocol that the microcontroller in this device uses to communicate with the EV. Essentially J1772 is a simple analog-control protocol that uses one data line (the pilot pin) to output a 1kHz pulse wave modulation (PWM) signal to the car. The duty cycle of the signal indicates to the car how much power it is allowed to draw from the house power line. The corresponding duty cycles are listed below:



AMPS	DUTY CYCLE
5	8.3%
15	25%
30	50%
40	66.6%
65	90%
80	96%

Table 1: J1772 Protocol PWM Duty Cycle and Corresponding Charge Rate

Peak voltage level of the PWM signal on the power pin secondarily is a status indicator of the car’s connection to the charger. The status criteria is listed below:



STATE	PILOT HIGH VOLTAGE	PILOT LOW VOLTAGE	FREQUENCY	RESISTANCE	DESCRIPTION
State A	12 V	N/A	DC	N/A	Not connected
State B	9 V	–12 V	1 kHz	2.74 kΩ	EV connected, ready to charge
State C	6 V	–12 V	1 kHz	882 Ω	EV charging
State D	3 V	–12 V	1 kHz	246 Ω	EV charging, ventilation required
State E	0 V	0 V	N/A	—	Error
State F	N/A	–12 V	N/A	—	Unknown error

Table 2: J1772 Protocol Pilot Pin Charge State Diagram

2.2 Related Work

The EVSE charger has been designed with different controllers many times before. One of its implementations, called OpenEVSE has been openly designed (all documentation has been made open source). Previous versions of devices designed for EVSE capability are listed below:

GWL Power EVSE kit:
https://www.ev-power.eu/Connectors-Cabling-1-1/EVSE-kit-for-EV-charging-station.html

Open EVSE charger
https://github.com/kortas87/simple-evse/wiki

TI’s level 1 and 2 wifi-enabled board
http://www.ti.com/tool/TIDC-EVSE-WIFI

Emotorwerks smart charger
https://emotorwerks.com/

Emotorwerks has implemented a design of this charger that meets all of the same exact requirements that our group aims to fix. However, the intended purpose of this project is not directly intended as a consumer product. The implementation of the hardware on this project is what sets this design apart from others. Design for this EVSE charger is intended to secondarily create a development board to make EVSE charging available for the arduino community and other budding embedded systems engineers,

Chapter 3: Problem Statement

This project has a series of requirements, constraints, and objectives necessary for its success. These are listed below.

3.1 Requirements

1) The charger must connect to a wireless network in order to regulate charge scheduling
2) Charger must be effective at scheduling charge times for the electric vehicle and must be able to fully charge an electric vehicle safely.
3) Charger must have an interactive interface capable of overriding scheduling
4) Charger must be capable of level 1 and level 2 charging for plug loads of up to 50A.

3.2 Constraints

1) The charger must adhere to J1772 communication standards
2) Charger hardware platform must be made directly integratable with the Future Electronics Creative board (based on Microsemi SmartFusion2), Arduino Due and ZigBee unit. (two separate designs will be developed) - phase 1.
3) Charger must use a custom wattmeter to measure its power consumption.

3.3 Objectives

1) Code will be implemented to control the charger, along with an implementation that uses an FPGA to control it almost entirely through hardware.
2) Charger design must be sealed and packaged neatly into a presentable package, ready for consumer use.

Chapter 4: Technical Approach

4.1 Structural Description

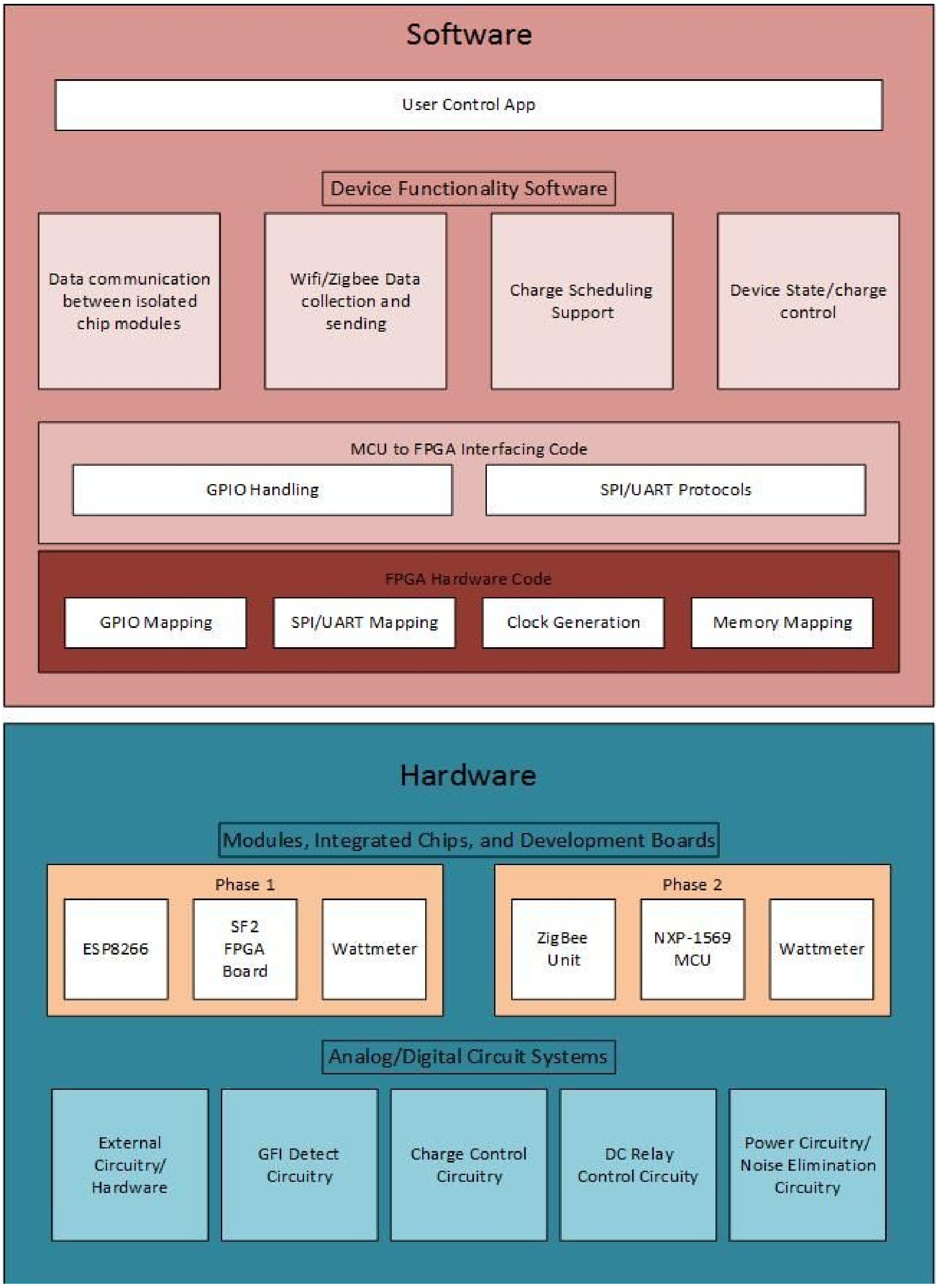

Software
User Control App
Device Functionality Software
Data communication between isolated chip modules
Wifi/Zigbee Data collection and sending
Charge Scheduling Support
Device State/charge control
MCU to FPGA Interfacing Code
GPIO Handling
SPI/UART Protocols
FPGA Hardware Code
GPIO Mapping
SPI/UART Mapping
Clock Generation
Memory Mapping
Hardware
Modules, Integrated Chips, and Development Boards
Phase 1
ESP8266
SF2 FPGA Board
Wattmeter
Phase 2
ZigBee Unit
NXP-1569 MCU
Wattmeter
Analog/Digital Circuit Systems
External Circuitry/ Hardware
GFI Detect Circuitry
Charge Control Circuitry
DC Relay Control Circuitry
Power Circuitry/ Noise Elimination Circuitry

Figure 1: Project Abstraction Diagram for Smart Charger

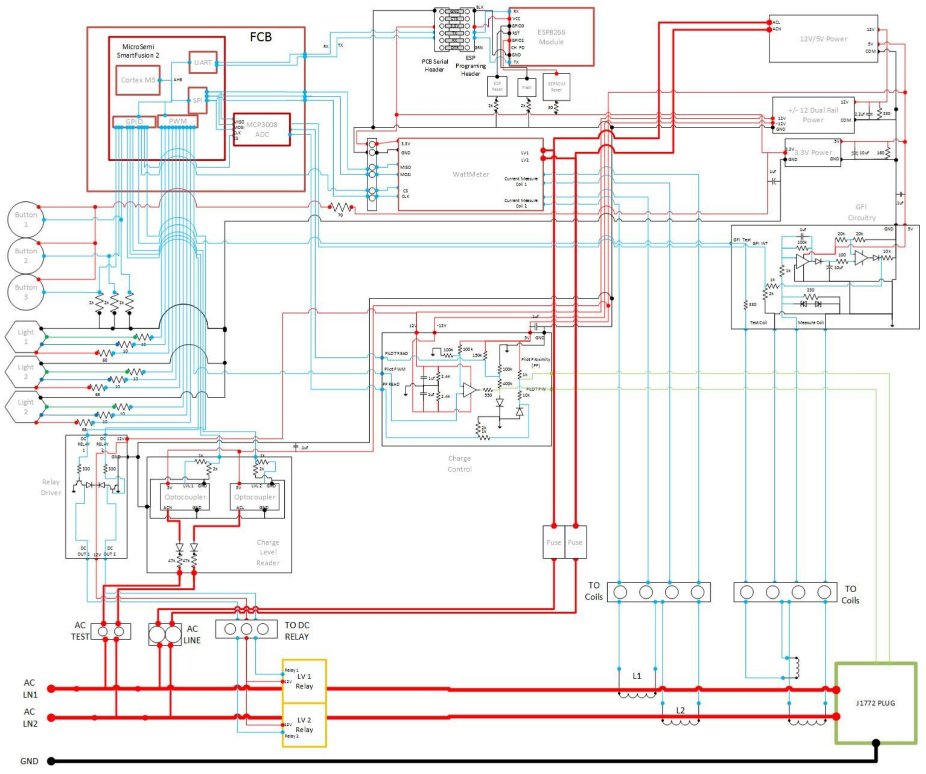


Figure 2: Detailed Electronic Abstraction EVSE Board

The schematic provided above is further clarification on the lowest level of hardware abstraction - analog/digital circuit systems. It has been added in order to illustrate the differences between the external hardware and internal hardware (internal meaning hardware that this team will be designing).

4.2 Flow Description

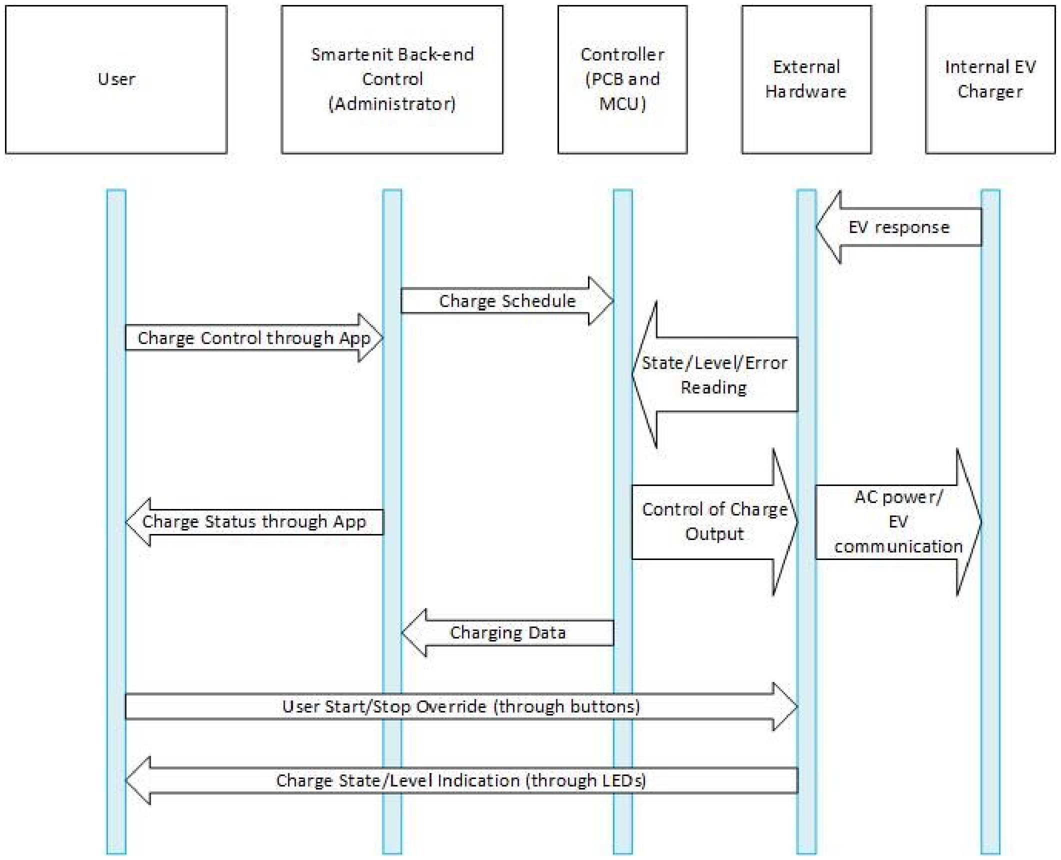

User
Smartenit Back-end Control (Administrator)
Controller (PCB and MCU)
External Hardware
Internal EV Charger
EV response
Charge Schedule
Charge Control through App
State/Level/Error Reading
Control of Charge Output
AC power/ EV communication
Charge Status through App
Charging Data
User Start/Stop Override (through buttons)
Charge State/Level Indication (through LEDs)

Figure 3: This sequence diagram encapsulates a typical interaction between a user and the smart charger device

4.3 Communication Description

The lowest level of abstraction of our system is the central power and communication hardware which is our PCB and the individual subcircuits on it. This includes the output LEDS, J1772 plug, DC relays, and GFCI circuitry. These circuits are all controlled by the GPIO pins of the microcontroller, which also interfaces with other devices on the PCB. The microcontroller is connected to the current sensor via SPI bus and the ESP8266 via the UART. On top of the hardware sits our first software abstraction layer, which includes the code used to program our FPGA fabric to allow it to link the SF2 IO pins to the ARM cortex microprocessor within it. Next is the higher level code which controls the inputs/outputs of each subsystem along with the state machine code that controls charging states. This layer also contains the code necessary to communicate with the ESP8266 and current sensor modules. Lastly, at the user application layer, either a mobile app controlled by the user sends charging information to the controller via Wifi/Zigbee, or Smartenit back-end code is sent to the controller to set charge scheduling.

4.4 User Experience

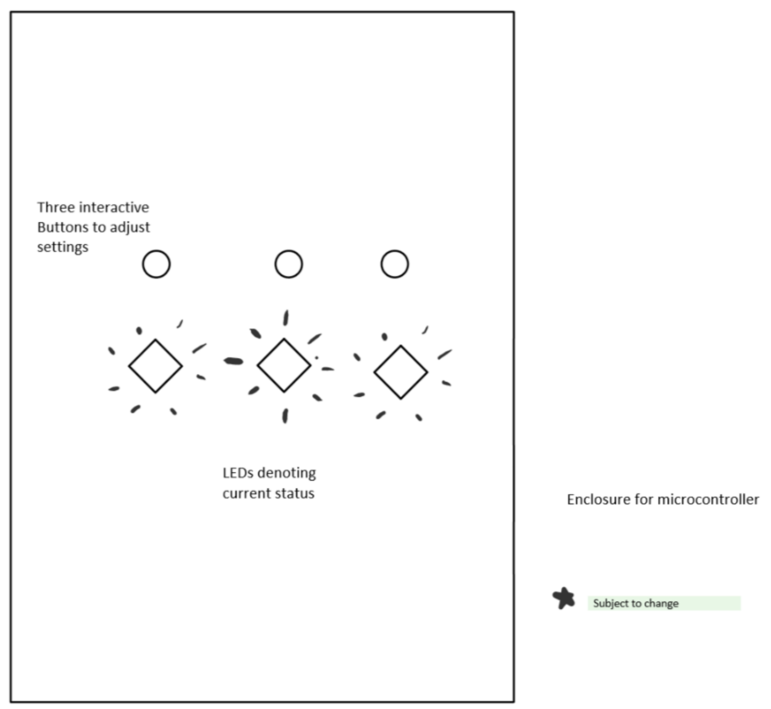

Three interactive Buttons to adjust settings
LEDs denoting current status
Enclosure for microcontroller
Subject to change

Figure 4: Crude User Interface Representation

Three pushbuttons will be available on the enclosure of the device that will allow the user to adjust various settings such as charging rate and charge time. The status LEDs show how much charge the car is obtaining. For example, if the LEDs are fully lit the car is being charged at maximum voltage. If the light is dim, the microcontroller is adjusting the voltage intake to be lower than maximum. When the second LED is on, it denotes that the charge is complete. The lights will also indicate which state of charging the car is in and whether any errors have occurred.

4.5 Technology Options

This project proposal has several implementations that require differing technologies as their root microprocessors. Phase 1 is built to interface with the Future Electronics Creative FPGA Board (SF2). This board is Arduino Due-compatible, making any board able to connect to it also compatible with Arduino Due. So both the Arduino Due and SF2 are options for charge control in our unit.

Phase 2 uses the NXP-1569 microprocessor module to control the workings of the charge controlling unit. It is a different implementation of the same system and will also require a specific PCB design to be used in the system.

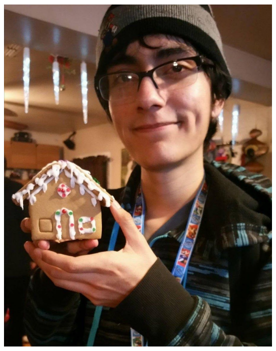
Both of these restrictions limit the technology options available to this project, however, this makes the task of determining the needed components much simpler. All other technology options are determined by what this design does not change from Chris Howel's Open EVSE design. The modifications of the Open EVSE previous design present many technology options from Mouser and Digikey (capacitors, resistors, potentiometers, and power sources). These technology options, however, are trivial in nature.

Chapter 5. Development Plan

5.1 Team Organization

Andrew Begey
ID: 46433784
Phone: 951-225-2366
Email: abegey@uci.edu
Skills: C++ and Arduino Programming, EAGLE PCB design, embedded systems design

Brandon Metcalf
ID: 78186959
Phone: (714)-6556930
Email: metcalfb@uci.edu
Skills: C++ and embedded systems programming, PCB design, embedded system device design, AutoCAD design, analog circuit design, digital circuit design



Luis Contreras
Major: Computer Engineering
ID: 93197465
Phone: (323)830-2988
Email: lecontr1@uci.edu
C, C#, Java, JavaScript, HTML, SQL, Python Programming Experience
Multithreaded application experience
Role: Software Design

Shermaine Dayot
ID:19055683
Phone: (213)880-0375
Email: sdayot@uci.edu
Major: Electrical Engineering
Skills: C, Python, HTML, Arduino, Raspberry Pi, electrical lab experience
Role: software development

5.2 Tasks and Responsibilities

This project is completed in two major phases. In the first phase (Phase 1), a custom EVSE controller board is designed for use with an FPGA is developed and tested. This will result in a board that is Arduino compatible, easy to test, use MQTT protocol connectivity over WiFi, and would be the world's first project Arduino shield for EVSE control. In the second phase (Phase 2), the team will use what was learned from phase 1 to design and build a residential "smart" zigbee protocol controlled EVSE solution that uses Smartenit's API. The team will demo and qualify the operation of this device.

Hardware Design and Implementation:
Andrew Begey
ID: 46433784
Phone: 951-225-2366
Email: abegey@uci.edu
Major Responsibilities: Develop PCB designs and coordinate with Brandon on building and testing hardware. Assisting with population and testing of the finished PCB boards.

Brandon Metcalf
ID: 78186959
Phone: (714)-6556930
Email: metcalfb@uci.edu
Major Responsibilities: Develop PCB designs and coordinate with Andy on building and testing hardware. Also responsible for heading off debugging strategies when integrating software components. Responsible for bridging any gaps between software work and hardware.

Software Development:
Luis Contreras
ID: 93197465
Phone: (323)830-2988
Email: lecontr1@uci.edu
Major Responsibilities: Assisting Shermaine in writing code for the communication between the SmartFusion2 device and ESP8266 wireless module. Responsible for leading development of the back end interface of the device

Shermaine Dayot
ID:19055683
Phone: (213)880-0375
Email: sdayot@uci.edu
Major Responsibilities: Assisting Luis in writing code for the communication between the SmartFusion2 device and ESP8266 wireless module.

5.3 Tools

Describe what tools you will use for different aspects of the project, including progress tracking, version control, collaborative authoring, code development, schematic capture, board layout, 3D and mechanical design, simulation, modeling, … If there is an aspect that can benefit from tools but you choose not to use any tools, explain why.

5.4 Timeline

First Quarter
- Week 1-2: Phase 1 start. Design Project Conceptual Layout
- Week 3-4: Develop board schematic with all components then design PCB layout
- Week 5-6: Confirm schematic constraints check prior to PCB fabrication

- Week 7: Generate bill of materials from PCB and send PCB to board house
- Week 8: Populate board and Perform pre-test of PCB
- Week 9: Design FPGA hardware and interface with PCB
- Week 10: test operation and control over wifi, using MQTT protocol and with Arduino
- Winter break: Interface FPGA with PCB/Test operation of the device/Finalize Results

Second Quarter

- Week 1-2: Phase 2 start; Develop board schematic and diagram for Zigbee support
- Week 3: Confirm schematic constraints
- Week 4: Generate bill of materials from PCB and send to board house
- Week 5: Populate Board and Perform pre-test on PCB
- Week 6: Confirm connectivity via Zigbee to Smartenit Home Area Network
- Week 7-8: Test and verify operation using Smartenit's application
- Week 9: Test and verify hardware override control
- Week 10: Finish debugging and finalize documentation

5.5 Resources

Proposed Budget:
Initial development of this project involves simulation and research on previous implementations of this board, however, the proceeding steps include sending our schematics to a board fab house for fabrication and populating our PCB with the required components. Testing and final project assembly will be performed onsite at CalPlug. These are addressed in the budget below.

Itemized Estimated Budget:

30 Amp J1772 EV charge connectors (2)	$346.72
Microsemi SmartFusion2 FPGA Demo Board	$99.95
Device Housing enclosures (2)	$84.00
Machining services	$45.00
12 Volt SPST DC relays (4)	$40.00
NEMA 50 Amp input cables (2)	$55.00
NXP JN5168 Demo Board	$187.50
PCB fabrication and assorted services	$200.00
High voltage wiring	$30.00

General electric components	$25.00
Assorted IC's	$25.00
Power Supplies	$40.00
Saleae Logic Analyzer	$219.00
Sigilent Oscilloscope	$259.00
Total Cost	**$1629.17**

include the budget, resources available from what sources, finance, where you plan to do the work, whether you have your own equipment or plan to use shared equipment or space, etc.

Resources

https://www.fueleconomy.gov/feg/evtech.shtml
https://www.technologyreview.com/s/518066/could-electric-cars-threaten-the-grid/