March 2021

Low-cost Hardware-In-the-Loop (HIL) Simulator for Simulation and Analysis of Embedded Systems with non-real-time Applications

Chanuka S. Elvitigala

Follow this and additional works at: https://digitalcommons.lsu.edu/gradschool_theses

Part of the Hardware Systems Commons

Recommended Citation

https://digitalcommons.lsu.edu/gradschool_theses/5262

This Thesis is brought to you for free and open access by the Graduate School at LSU Digital Commons. It has been accepted for inclusion in LSU Master's Theses by an authorized graduate school editor of LSU Digital Commons. For more information, please contact gradetd@lsu.edu.
LOW-COST HARDWARE-IN-THE-LOOP (HIL) SIMULATOR FOR SIMULATION AND ANALYSIS OF EMBEDDED SYSTEMS WITH NON-REAL-TIME APPLICATIONS

A Thesis

Submitted to the Graduate Faculty of the Louisiana State University and Agriculture and Mechanical College in partial fulfillment of the requirements for the degree of Master of Science

in

The Department of Electrical and Computer Engineering

by

Chanuka Sandaru Elvitigala
B.Sc. Information Technology (Hons.), University of Moratuwa, Sri Lanka 2018
May 2021
To my family and teachers who believed that I can do good and friends who helped in different ways.
Acknowledgment

I take pleasure in submitting herewith the report on “Low-cost Hardware-In-the-Loop (HIL) Simulator for Simulation and Analysis of Embedded Systems with non-real Time Applications” in partial fulfillment of the requirements for the degree of Master of Science in Electrical and Computer Engineering.

I am grateful to my adviser Dr. Gerald Baumgartner for his tremendous support in every way. The knowledge that I have gained from Dr. Baumgartner was tremendous and his words of wisdom helped me to make many of my decisions on thesis and life. His willingness and support led to the success of this project and his guidance on how to convey these findings to society led to fine-tuning the project. He was patient with my silly mistakes and faults and I’m truly grateful for that.

I would also like to thank my committee members Dr. Doris Carver and Dr. Feng Chen for graciously agreeing to be on my thesis committee not only they are my committee members but also they were my lecturers in several subjects.

There were times that life was really tough for me and my Family who kept me strong. They showed me the path to be relived and healed me I could not pen the gratitude I have towards them. Because without them I’m not the person I am today. I also thank the teachers who tough me not only the subjects but life as well. I also thank the Sri Lankan student community at Baton Rouge for their tremendous support.

I thank my roommate Mr. Malith Chathuranga who helped me when I was busy with work and who taught me to make nice graphics used for this thesis.
# Table Of Contents

Acknowledgment .................................................................................. iii

Table Of Contents .................................................................................. iv

Abstract .................................................................................................. vi

1. Introduction .......................................................................................... 1
   1.1. What is an Embedded System? ....................................................... 1
   1.2. What is Hardware-In-the-Loop (HIL) Simulation? ......................... 2
   1.3. What is the use of HIL for slower Embedded Systems? ................... 3
   1.4. Proposed system and approach ...................................................... 4

2. Related Work ......................................................................................... 5

3. Technology Review ............................................................................... 9
   3.1. GPIO (General Purpose Input Output) ........................................... 9
   3.2. Multiplexer .................................................................................... 11
   3.3. Demultiplexer ............................................................................... 11
   3.4. PWM (Pulse Width Modulation) ................................................... 12
   3.5. ADC (Analog to Digital Conversion) ............................................. 12
   3.6. UART or SCI ................................................................................ 13
   3.7. I²C communication ...................................................................... 14
   3.8. NodeMCU ...................................................................................... 17
   3.9. Arduino Mega ............................................................................... 18
   3.10. Texas Instruments TMS320 DSP on Spectrum Digitals eZdsp F28335 .. 19
   3.11. Real Time Clock (RTC) with BQ32002D ....................................... 19
   3.12. Analog to Digital Converter with ADS7828 .................................. 24
   3.13. 4-Bit 1-of-2 FET Multiplexer/Demultiplexer with SN74CBT3257C .... 26

4. Approach ............................................................................................... 28
   4.1. Device Design ............................................................................... 28
   4.2. Communication Design ............................................................... 36
   4.3. Playbook Implementation ............................................................. 40
   4.4. Software Design .......................................................................... 40
   4.5. Cost for Devices .......................................................................... 40
   4.6. Example Setup ............................................................................. 41

5. Data Explained ...................................................................................... 43
   5.1. Performance of GPIO pins ............................................................ 43
   5.2. Performance of ADC in Arduino, TMS320 and Peripheral Device ...... 44
   5.3. Performance of PWM ................................................................. 45
Abstract

Hardware-In-the-Loop (HIL) simulation is an approach that is used for embedded systems testing which was popular in the early days of embedded systems. This technology was wrapped and was stored away for a long time. One of the main reasons is that the less use of embedded systems in society. Recent developments in embedded systems, data science, and control systems lead to a focus on testing these systems. The earliest method of testing these systems was by doing destructive testing which is a waste of money, resources, and time. The HIL provided a safe and economical way of testing the embedded systems while in development. The major problem is that HIL systems are really expensive and normal users, hobbyists, and students cannot afford them. Usually, they are the group who are most effectively burning up electrical systems while testing and learning on systems.

There are many types of research on HIL and its use in many applications. Mostly they are on an industry level or highly funded researches. The normal users or our targeted users do not have access to such advanced equipment. The HIL is used recently for rapid designing of embedded systems for nuclear power plants and vehicles. This system is simulating the external environment and sensors and actuators to the embedded system which make them “feel like” they are in the actual environment. By this the embedded systems responses and state changes are monitored which help the developers to debug quickly than conventional methods.

Our main target is to create a low-cost device but which can perform nearly good as the industry-level simulators. This helps these users to create embedded systems in very low cost and test them rapidly than their conventional method of destructive testing. This saves the users resources, money, and time. As an example a user can create a motor control system and test the embedded systems (motor control systems main controlling unit) proper functioning while the user code the system. This also prevents users making mistakes and burning the electrical equipment while testing (destructive testing). The users can make more accurate and complex systems with lesser effort than the previous manual process.

The start of this journey is to create a system that can detect slow processing elements and their changes. We have used 4 System designs that are different from one another by several small factors and tested them using several conditions. We also have implemented a communication protocol which is using the wireless network. This is enabling the users to remotely access and control these simulators. The controllers we have used are Arduino Mega, Node MCU, and TSM320 DSP.

The devices we have designed were undergone different test strategies and criteria to be selected. The main concern was to have a device with lowest cost as possible within little change in hardware elements and with highest precision and efficiency of monitoring. We have selected the Device two which is Arduino Mega which is connected to a Multiplexer switch and WiFi module that we have designed.

This development is done to show that the target of making a HIL is achievable for slow systems at a very lower cost. This also proves that this could help normal users, hobbyists, and students with their work in embedded systems without damaging the hardware and in a lower time span.
1. Introduction

1.1. What is an Embedded System?

The Hardware-In-the-Loop Simulation or shorter HIL is the main discussed topic in this document. Before talking about the HIL and its uses the attention towards the embedded systems have to be made. These small chips have changed the course of technology and paved a new way for humankind. But it is still unnoticed and working without getting the credit it deserves.

Embedded systems are the systems that are more job-specific or task-specific if we compare them with general-purpose computers. An embedded system can be a simple isolated system to a small element in a larger system. In these systems, a normal connection to the outside world is made and the system is acting based upon the inputs from the environment. The system is comprised of the main controller, sensors that take input from the environment, and actuators that affect or reflect the reaction of the system to the outside world [1, 2, 3].

There are many different kinds of controllers that are appearing in the market. If we consider the manufacturers they have an array of different IC series designed for several different purposes. If we consider the embedded systems they can range from reading the temperature in an environment to handling a motor as a motor driver controller to even handling communication between the engine and controlling the engine in a vehicle.

When we dive deep into the embedded system design we see that some designs are time-critical or as per the proper technical terms real-time. But some of them are not real-time or bounded by the time constraints but simply acting as a normal controller and their change of events are noticeable to the human observers. In this document, we also would go on and talk more about these comparatively slower systems and how our project is going to tackle some of the problems that are associated with these systems [1].

As these systems have a wide variety of manufacturers the way they can be programmed and eco-systems of their programming environment can change dramatically or sometimes can be extremely similar. The programming of these systems and using another controller which is from a different vendor is uncharted territory with many compromises that have to be dealt with the system design. So mostly systems designers tend to use IC’s that are manufactured and designed to last long and with many customer bases and feedback towards any issues found.

Due to these problems, early systems designers were a bit hesitant to change to another controller vendor and stayed loyal to the vendors they are accustomed to but as the information age boomed many designers came out of their comfort zones and tried new controllers. With this, each controller series had an enormous amount of support both from the vendor and third-party open source community which made the embedded systems more popular and go-to application methodology in systems design [4].
1.2. What is Hardware-In-the-Loop (HIL) Simulation?

Hardware-In-the-Loop simulation is a methodology that is designed to test embedded systems. This is used in most of the critical systems or expensive systems before the actual use of the embedded system within the device. Consider a nuclear power plant or any other critical system. This system might contain thousands of inputs and outputs which will control and maintain the plant. If there is a problem in the code that is used in the controller this will make devastating effects on the system and environment. The software designers for these embedded systems need an accurate method to test these systems as the system has many actuators and sensors that work in a given environment and modeling them will lead to a complex mathematical model [1, 5].

The HIL systems have to deal with dynamic systems and their simulations. Most of the industry-level applications are real-time to mitigate these risks and address the challenges they have to design systems that can identify real-time inputs and give outputs to the controllers while attending to complex mathematical models. This makes these systems expensive and complex to handle so a specialist should be there to handle these systems.

The HIL system is an emulation of the plant (The target system where the embedded system is going to be placed) and it emulates the inputs (sensors) and outputs (actuators) some of them also aim at developing the system to be able to emulate the effects on the environment [5].

The metric of development and testing of a given embedded system will be based on cost, duration, safety, and feasibility. Keeping these goals in mind embedded systems developers try to achieve the given requirements of the system.

The HIL simulators provide the following advantages when they are being used in the process of developing embedded systems.

1. To improve the quality of testing.
   This happens when the embedded system is tested outside its operational parameters because some of the plants have their limitations on testing and if the designers want to test in more extreme conditions than they have expected and if they include any new scenarios that are highly unlikely to occur in the system they can test it using the HIL.

2. When there are tight schedules to be met in development.
   Many developments related to embedded systems are extremely time-oriented and the product must be delivered in a tight schedule so without actually planting the embedded system in the real environment the designers can test the systems easily on a test bench.

3. When there is a safety critical application.
   When developing the resemblance of the plant to the simulator is high and it is cost-efficient to use a simulator than a real system the developers tend to use HIL for testing.
4. When humans are involved in the testing (Human In the Loop).

When a plant uses human interaction or feedback to determine a task it is easy to use a human with a HIL simulator to get the input rather than using a human in the plant.

1.3. What is the use of HIL for slower Embedded Systems?

The slower embedded systems which are relative and have distinctive state changes to humans use more human testing rather than the use of HIL simulators. This has been done as a practice in the industry quite often. Our main focus is using them for personal use and educational purposes than the industry environment [5].

The first use is that the students and personal users have a way of testing their code more efficiently rather than using manual labor. As this electrical equipment often does not have a console to identify code errors or the errors caused by the embedded system code they have to use alternative measures such as using LED lights or any other indicators to identify each part of the code or the errors of code. This is synonymous with printing console output in the code to verify the code execution path and the values in software development. Although people use this technique as an effective method in some cases it is not the case for the embedded systems because the indicators sometimes can indicate code execution path but it is hard to decode the values virtually held by variables. This also takes time. If we use a HIL simulator we can get the output of the system according to the input we can see the execution path easily but still, the heap or memory stack can not be seen in the HIL but this can be identified through debugger tools which are included in most controllers with JTAG (Joint Test Action Group) interface. This interface is a debugging and code transfer mechanism implemented to unify all the code upload and debugging related to the embedded systems [6].

As described previously the manual testing can take time. especially if the people are using indicators the electrical wiring should be done specifically for those instances using HIL will not only reduce this burden but also can be helpful to test embedded systems without using the plant. This can save a lot of time at the development end and testing end. Some of the systems even they are slowly meant to work on a specific frequency or time delay using these indicators and adding them to the system can add additional time to the controller which make the timing inaccurate in the system and create an extra burden to the controllers which is a very small controller compared to the general-purpose computers that are available. This extra wiring also needs pin outputs and if the systems all pins or needed pins are engaged in systems tasks they cannot be freed up for the task of indication. This also might lead to removing some of the devices from the system and use indicators for testing which later on lead to full integration testing within the plant to see the accuracy of the code. Using HIL will free the designers from this burden as there is no need for extra pins and ports for indication [7].

The most important aspect of designing the HIL for these kinds of applications is to reduce the amount of rework and hardware corruption. As said earlier this system is made specifically for personal use and for students. The biggest barrier for the target community to enter hardware-based design is the cost of electrical devices used. As an example, some
of the motors can cost up to 1000 dollars depending on their use normally an individual or a student has to bear this cost and get this equipment for their testing. The most common case in electrical design is electrical shorting and wrong handling of power which can lead to burning of equipment. If the devices are costly these people cannot afford to burn the devices and if in students’ case they have depleted a costly resource to the next student group. Using HIL make sure that code is safe to be used in the plant given and this keeps the designers’ mind at ease of destroying some equipment due to minor errors in the code.

Another most important aspect of this is remote connection. Most of the electronics labs can be conducted online if the simulator or the target device can be accessed online. This will save the students time and energy and make the lab more engaging.

By investigating into above claims it seems that HIL is a great tool for not only the industry but for the students and individuals as well which saves their money, time, and effort when they are creating an embedded system-based design.

1.4. Proposed system and approach

The HIL system that we are proposing is the first step of achieving the target of creating a low-cost HIL for students and individuals. This in return is also a proof of concept that this kind of system can be designed at a low cost. The main target of this document is to find a mechanism to simulate and record information through digital signals in a slower embedded system (taking minimally about 1 second to transfer from one state to another). This includes various problems on how to record the communication channels in the system and it is considered as further work for the sake of this document. The main purpose will be the study of the viability of using HIL to monitor slower embedded systems Digital I/O, Interrupts, ADC, and PWM capabilities. The HIL system that we are proposing should be able to deal with any embedded system designed by any vendor. As an example, the system should be able to assist Arduino-based microcontroller or Texas Instruments microcontroller despite their difference of manufacturer, compiler, or the instruction set. The main components being the WiFi communicator with Time stamp generation ability, data collection software, and simulator for the system. The 4 device implementations are,

1. Device 1 : Arduino Mega with Switch and Analog reading Module
2. Device 2 : Arduino Mega with Switch only
3. Device 3 : TSM320 with Switch and Analog reading Module
4. Device 4 : TSM320 with Switch only

One of the device from above will be selected based upon their performance during the testing. This system will help users to analyze and simulate environment to non-real time applications based embedded systems. This also helps to analyze the code of the system using black box or white box testing capabilities. The developers can change the algorithm on the fly which will break the normal structure of waterfall model in embedded system design.
2. Related Work

The discussion on the related work can be kicked off by diving into the predecessors of the current project which is done under the supervision of Dr. Gerald Baumgartner. The work on “A Virtual DSP System for Design Instruction of Power Converters” laid the groundwork to initiate the project on creating an affordable and remotely connectable simulator for the students to learn electronics and embedded systems. The main problem that is identified is that the cost of many electrical devices and their availability to a large set of students. The immense complexity and sensitivity of the equipment made this project more crucial in the long run. The main topic discussed in this paper is using TSM320 DSP as a HIL to simulate the resonant power converters. These systems are extremely time-dependent and prone to malfunction if there is a wrong application of power or input to the circuit. This also affects the device attached to the power converter such as AC motors etc. which leads to burning of the system or serious malfunction. The problem is that as these systems are extremely vulnerable to errors and they are costly so it is hard to use them as a teaching tool in the classroom environment. To mitigate this the HIL system which is using TSM320 DSP was proposed. The simulation is done mainly from the software in this project and this is adopted to our current design but using actual electrical signals when simulating to the embedded system is considered. The preference over software simulation is that in real electrical inputs some imperfections occur due to external EM radiation. Our system can partially create this kind of environment for testing [8].

The paper “A Virtual Embedded Systems Testbed for Instruction and Design” is focusing on designing a HIL which is having a controller connected to a PC’s parallel port and the PC being the simulator of external devices. The PC hosts HTML based web server which can be used for the users' interactive environment. So the distance learning can also be performed through this system. The system occupies the XML as the data transfer format in this setup. The main idea from this system is taken to the current project where we use a network-based server that is open to the network and simulator is separated from the PC and special hardware is used for this purpose. This also fulfills the lack of using analog signals in the given system. The given system is used for educational purposes and distance learning this is a breakthrough design according to the writer’s perspective at the time where distance learning was not popular [9].

The next publication is on “An Integrated Virtual Learning System for the Development of Motor Drive Systems”. This is taking the virtual learning systems further ahead by introducing a GUI-based front end and DSP-based HIL system. The main focus is on motor controlling which creates more attention towards the implementation of the PWM simulation necessary for the circuits. The researchers are focusing on the 3 phase brushless DC (BLDC) motor control applications. The use of object-oriented concepts in the design is a major leap forward in this project series as OOP concepts are widely distanced from embedded system designs due to several reasons. The major one begins that the embedded systems are working as state machines and this can be achieved by functional programming. The program also provides the student with necessary feedback and the students can do the simulations leisurely and grasp the idea of controlling BLDC in their own time. Being not a
constraint to a lab environment and having the accessibility to this equipment (simulations) make the learning more fun and interactive to the students [10].

The paper “Giotto: A Time-Triggered Language for Embedded Programming” is describing a model that is abstract and can be used for the implementation of embedded control systems with hard real-time constraints. This implementation is comprised of time-triggered events and schedulers. This is a hardware-independent design so it is very suitable for any embedded application in any environment. The writers also emphasize that this system is suitable for safety-critical applications. This is a software model so the ideal conditions as discussed above can be seen despite having inconsistent signals that we see in the normal environments [11].

The publication “Chrona’s Validator tool suite: filling the gap between conventional software-in-the-loop and hardware-in-the-loop simulation environments” discusses the importance of the HIL approach. According to the authors, the HIL can have accurate simulations than the SIL (Software In The Loop simulations). The simulator is powerful and describes that it will fill in the gap between the verification and validation process in the SIL and HIL. This system is also helping the co-simulation between other available simulators. The problem with this system is that even though it is cheap from the industrial point of view it is really expensive for personal or educational purposes. Another point to be considered is that to use this kind of tool it takes an exhausting training process [12].

In “The Embedded Machine: Predictable, Portable Real-Time Code” The designers are introducing an embedded machine that sits between a platform-independent compiler which turns the code into a code called E code and can be used for supervision of the timing in the system. The E code can generate the output depending on the timing and input given. This can be used to test the actions of the embedded system. The next phase will be a platform-dependent compiler that checks the E code for time safety an attribute determined by platform performance and platform utilization which enables the timely execution [13].

The paper “Hardware-in-the-Loop Simulation of Grid-tied Converter for Unity Power Factor Operation” is discussing about a pressing issue. When a renewable source is connected to the grid the power factor of the connected component should be equal to the grid otherwise there are some problems like heating up wires and damages to the system etc. In this paper connecting a renewable energy source to the grid is considered and HIL simulator is used to analyze a control technique for a grid tied converter so as to have a unity power factor operation in the system. The paper consists of many tests related to the issues and performance of this operation that is described in the paper [14].

The paper “A Novel Interface Modeling Method of Power Hardware-in-the-loop Simulation for MMC-HVDC System Based on ITM and DIM Switching” is discussing about the difference between ideal transformer model (ITM) method and damping impedance method (DIM). This comparison is done through power HIL which is a real time power grid simulation environment [15].

The paper on “Design of Hardware-in-the-loop Simulation System based on RTX and FlightSim™” is aiming on converting general purpose windows based computers into a real-time simulator. Then by using the help of FlightSim which is a strong mathematical simula-
tion platform they are going to create a HIL simulator. The RTX (Real-Time Extension) is the expansion that they are going to use to convert a windows based machine into a real-time simulation environment [16].

The publication “Using Hardware-in-the-Loop Simulation Platform for Empirical Testing of Photovoltaic System” is focusing on the simulation of the photovoltaic power generations and their support to the grid. The simulations in this project are mainly done through the use of the HIL simulation [17].

Another research is “The Hardware In-Loop Simulation System Based on SUMO for Autonomous Optimization Control Algorithm at Intersection” which has paid attention to the traffic controlling problem where the developers used the SUMO (Simulation of Urban Mobility). The interface of the SUMO is connected to the HIL that is designed by the researchers and used microwave radar detectors as the input to the system [18].

The paper “MPPT Test Based on Hardware-in-the-Loop Simulation Platform of Photovoltaic Systems” is a document which is related to photo voltaic grid-connected inverter system. The HIL’s connections are connected to the physical controller of this system and simulated various environmental conditions to the device to analyze the performance [19].

In “A Hardware-in-the-loop Simulation System of Deep Space Autonomous Navigation Based on Angle and Velocity Measurement” discuss a complex and novel approach for deep space navigation which is base on angle and velocity measurements. The HIL is used to simulate different environments and sensor data for the space vehicle to analyze its reaction to each case. The experiment data gathered in the process is used to enhance the algorithm used in the space vehicle and other navigation systems [20].

In the “A hardware-in-the-loop simulation method for the evaluation of flight control systems” their flight simulation system has been attached to the HIL system. The HILs evaluation of the FCS is used and compared [21].

Another publication is “Control Hardware-in-the-Loop Simulation on Fast Frequency Response of Energy Storage System Equipped with Advanced Frequency Detection Algorithm” which is working on fast frequency response (FFR) of the energy storage system (ESS). This is used to find the effects of the frequency of energy generation by renewable energy and their storing systems. The HIL is used for the simulations and collects the data for the data analysis [22].

The publication “A Scheme to Improve the Stability and Accuracy of Power Hardware-in-the-Loop Simulation” is considering the improvements related to power HIL simulation algorithms. Researches in this publication have found that the transportation and communication delay creates instability and inaccuracy in PHIL simulations. This made them synchronize the output signal and promote stability and accuracy through a time compensation scheme proposed by them for the PHIL simulations [23].

The “Research on Hardware-in-loop Simulation Platform of Anti-swing Control for Helicopter Suspension” is describing the HIL simulator which is design to simulate anti-swing control for helicopters. A 6-DOF robot arm is used to simulate the slung helicopter. This paper concludes that the anti swing control can be simulated through the HIL [24].

The paper “Real-Time Hardware-in-the-Loop Simulation and Control of Totem Pole
PFC Converter” is describing how to simulate a totem-pole power factor correction (PFC) circuit. The power factor correction is used to reduce the amount of reactive power created by circuits. The totem pole is a new type of circuit that reduces the number of semiconductors from three to two. According to the authors, the challenges like current spikes that occur at zero crossings, DC component in an AC main current, and AC voltage drop handling are investigated in this paper [25].

In “An Effective Power Hardware-in-the-Loop System for the Simulation Testing of an Energy Management System of a Nearly Zero Energy Building Microgrid” the paper describes a laboratory test environment for evaluation for the performance of nearly zero-energy building (nZEB) microgrid using PHIL simulation. The laboratory has a 10kWh lithium-ion battery array and is used as the power storage mechanism [26].

The paper “Incorporating Hardware-in-the-Loop Simulation into Object-Process Methodology” is trying to in-cooperate HIL to Model-based systems engineering (MBSE) using the Object Process Methodology (OPM). They are also considering a cloud-based approach to this design in the latter part of the publication [27].
3. Technology Review

3.1. GPIO (General Purpose Input Output)

The GPIO is the most general way of describing the work in a microcontroller or processor. This is a pin that can be set to high (5V in Arduino) and low (0V in Arduino) to indicate a signal. Also, the pin can be rearranged to be a receiver of the signal and the signal can be reflected in a data registry inside the controller for further use.

![GPIO Circuit Diagram](image)

Figure 1. Example of GPIO circuit inside a controller [28]

The Figure 1 is showing a general circuit view of the GPIO pin setup. In the circuit, the PINMODE determines whether a pull-up resistor needs to be applied to the pin as in some inputs the input voltage if not have been pulled up will be in a floating state which is a High impedance state. The floating state creates random voltages appearing in the pin which will be garbage values. The PINDIR selects whether the pin should be an input or an output. The diodes are given for the protection of the pin from low current transient events (sudden oscillation of voltages or burst of voltages) [2].

**GPIO Hysteresis**

The GPIO pins take inputs from the outside world when the voltage provided to these pins changes there are some sudden fluctuations of the voltages where the PIN values can abruptly change without any actual information on them. The Figure 2 shows how to mitigate this
risk. The hardware designers use an RC (Resistor Capacitor) filter near the pin so that these unwanted frequencies are grounded. But since there can be sudden voltage changes the designers of the controllers keep a high threshold where if the voltage goes higher than this value will make the pin value High and a low voltage threshold where if the voltage goes below this voltage will make the pin state to low. The gap between the two thresholds is called a dead-band or gray area of voltages and this can be shown by Figure 3 [29].

![Figure 2. The RC circuit attached to the microcontroller pin [30]](image)

![Figure 3. The voltage hysteresis [31]](image)

**GPIO Interrupt pins**

The GPIO has special pins designed to receive interrupts. As standard software-based interrupts, these lower-level pins will be notified of any changes to the pin is occurred. Even though many controller developers use different mechanisms to implement this process the use is more or less the same.
When a pin change occurs the special hardware will interrupt the controllers’ work this happens only if we configure to receive the GPIO interrupt in a specific pin. After notifying the controller the controller will go to the ISR (Interrupt Service Routine) related to the pin and perform the task. Most of the controllers carry higher weight on the GPIO interrupt and have higher precedence in the ISR vector tables [29].

### 3.2. Multiplexer

The multiplexer or the “Mux” in shortened form is a digitally controlled switch in simple terms. In this design, multiple inputs can be switched between one output line to make a connection. The control pins are used to switch the input lines with a single output line. The Figure 4 shows an example setup for that purpose [3].

![Multiplexer Diagram](image)

**Figure 4. A multiplexer [32]**

### 3.3. Demultiplexer

A demultiplexer is the opposite of the multiplexer it takes a single input line and connects it to several output lines. This is also a switch and has selection lines to switch the connection.
This device also has a short name which is “DeMux” and this device is also a primary choice when there is an application for switching. The Figure 5 shows an example setup for that purpose [3].

![Diagram of a demultiplexer](image)

Figure 5. A demultiplexer [33]

3.4. PWM (Pulse Width Modulation)

The pulse width modulation is another application of the GPIO’s variant wherein this application a wave is generated. This wave is a digital wave with a specific “on time” (time, where the output, is high) which is called a duty cycle. If in one cycle the pulse if High for 50% of the time then the duty cycle is 50%. The reference device gets the 50% of actual voltage that it is intended to be receiving. As an example, if the duty cycle is 20% in an application and the pin’s voltage high value is 5V then the voltage that the device connected to the pin receives is 1V. This has applications in controlling motors speed and servo motor control with other applications like multicolored LED’s. In multicolored LEDs, the color is determined by the voltage received to the LED. Another famous application of this is its use as a dimmer [3].

3.5. ADC (Analog to Digital Conversion)

This will convert an analog signal to a digital signal. This can also be seen as converting from continuous form to discreet form. The main key for the conversions is sample rate and resolution. If the analog signal is translated into a digital form the sample rate is very important. According to the Nyquist theorem for a signal, at least the sampling rate should be double the size of the input signal. This means to sample rate for 50Hz should be at least
100Hz. The resolution is the next important thing in the ADC here the number of steps a voltage divided is considered. This also keeps the almost continuity and smoothness of the data collected. as an example 10 bit resolution can have 1024 steps which almost make a change of 5mV detectable. An example for such operation is shown in Figure 6 [3].

![Figure 6. An example of AD conversion](image)

### 3.6. UART or SCI

UART (Universal asynchronous receiver-transmitter) and SCI (Serial Communications Interface) mean the same communication method. Most people also label this as RS 232 communication. This communication method uses two wires and they are named RX (Receiver) and TX (Transmitter). This uses FIFO registers and as the transmitter and receiver do not share a clock they have to define the data rate. A high level diagram of UART communication is shown in Figure 7. This is called the baud rate of the communication. In addition to this the transfer data frame structure has to be negotiated before the transmission this includes the number of bits, use of start bit and stop bit, and checksum [3].
3.7. **I²C communication**

The I²C communication or inter-integrated circuit protocol has a rich history and ecosystem of devices that support and use its full potential. This is designed for short-range communication but it need only two wires to maintain the communication channel. Example setup of such communication is show in Figure 8 [36]
This communication methodology was originally proposed by Philips for its various chips series starting from 1982. This originally had a 7-bit address block which made the total number of devices that can be connected to 112 with several reserved addresses. In 1992 the specifications are published with a 10-bit address and 400kHz fast mode which can go up to 3.4 Mbps. But this was not enough for the industry and their standpoint so they have designed several other flavors of this protocol which can go from 1MHz to 5MHz speeds. The microcontroller community has stayed on the 400kHz range as it is convenient and well settles with other peripheral devices that are connected to the system. As we are also dealing with many microcontrollers that are widely used we tend to be using the “Vanilla” protocol that was published. Intel also introduced a variant of its own called System Management Bus (SMBus) with a tightly controlled format of data transfer.

There are many other data transfer protocols which are Serial Peripheral Interface (SPI) and Universal Asynchronous Receiver-Transmitter (UART) widely used in the microcontroller systems but what makes \( \text{I}^2\text{C} \) unique is that it has the best of both protocols. The following will point out the major facts which contributed to use \( \text{I}^2\text{C} \) over others.

1. UART Communication

(a) Before the beginning of the communication, the two devices should agree on the speed of the data rate or they should have the same data clock rate otherwise the data will be destroyed in one end.

(b) The communication inherits a hardware overhead which needs at least one start bit and end bit for every 8 bits which makes the total data send 10 bits and it can slow the communication.

(c) Only two devices can communicate otherwise the bus contention occurs where two or more devices try to use the data bus at the same time this can not only

![Diagram of I2C configuration](image)
damage the communication but also the hardware if the communication device’s hardware is not properly designed.

(d) There is not theoretical data limit as the data speed depends on the clock speed that the hardware has.

(e) Only two wires are required to communicate which makes the electrical designs easier and more compact.

2. SPI Communication

(a) Many pins are required to achieve the communication where 4 of them totally required for one device.

(b) Only one communication controller (controller) can be in one channel.

(c) Inherited with a good data rate which goes upto 10 MHz in full duplex communication.

(d) The communication hardware can be implemented using simple elements like shift registers.

3. I²C Communication

(a) Only two wires are involved which are Data (SDA) and Clock (SCL) in the communication.

(b) Communication can freely use 1008 peripheral devices.

(c) The communication can occupy more than one controller (controller).

(d) The data rate sits between UART and SPI.

(e) The communication has some overhead as for every 8 bit there is 1 bit which ensures ACK/NACK in the communication.

(f) Hardware implementation is complex than SPI but less than UART.

I²C has SCL which is the clock signal and SDA data signal. The clock signal is generated by the bus controller at that specific time. Sometimes given peripheral devices are forced to clock low at times to delay the controller sending more data and this is called “clock stretching”.

The I²C uses an open drain where they can pull the corresponding signal line low but cannot drive high this will protect the devices in the data bus from bus contention. This requires the data bus to have a pull-up resistor as a rule of thumb for short distances and a low number of devices the pull-up resistance value is 4.7 kΩ. To handle longer distances we can use signal extenders to the bus.

In the communication, the messages are broken into two parts where an address frame and a data frame are present. This can be shown by the diagram given [3, 7].
The basic communication of 7-bit addressing is shown in Figure 9 where the signals pattern is shown that is effective in the communication line. As a side note when a peripheral device is read some of the devices have the capability that is built in to increment the reader to the next available register rather than notifying the register to be read. The I²C library Wire in Arduino has the device address as the lower 7 bits and the MSB is zero. The uses in Texas Instruments DSP is different from the Arduino Megas use as it has much additional functionality included which is like a 10-bit address. As a side note on using this communication in Texas instruments devices, it is best to use FIFO buffer with the communications [39].

3.8. NodeMCU

NodeMCU is a product of many different controllers together the device mainly focuses on enhancing the capabilities of the ESP8266. The device is armed with the ESP8266 as the core processor which is also an IoT enabler that has WiFi access and a Lua-based on-module flash-based SPIFFS system. The device uses C language as the higher-level programming language that supports the system. Lua is a lightweight embedded scripting language. This is used as the support for the SPIFFS file system which is intended for SPI NOR flash devices. This simply means that the flash chip which is used is NOR Flash and the communication is done through the SPI communication. This file system has different capabilities such as,

1. Small (embedded) targets, sparse RAM without heap.
2. Only big areas of data (blocks) can be erased.
3. An erase will reset all bits in block to ones.
4. Writing pulls one to zeroes.
5. Zeroes can only be pulled to ones by erase.
6. Uses statically sized ram buffers, independent of number of files.
7. Posix-like api: open, close, read, write, seek, stat, etc.
8. It can run on any NOR flash, not only SPI flash - theoretically also on embedded flash of a microprocessor.

9. Multiple spiffs configurations can run on same target - and even on same SPI flash device.

10. Implements static wear leveling.

11. Built in file system consistency checks.

12. Highly configurable.

This system is using the ESP8266 WiFi capabilities through AT commands. The AT commands are made for issuing commands to modems and GSM phones which still in use up to this day. The ESP8266 also holds a great reputation for usages in the IoT field as it is using WiFi communication and can send data directly to an online platform without the intervention of a middle machine.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Certification</td>
<td>Wi-Fi Alliance</td>
</tr>
<tr>
<td>Protocols</td>
<td>802.11 b/g/n (HT20)</td>
</tr>
<tr>
<td>CPU</td>
<td>Tensilica L106 32-bit processor</td>
</tr>
<tr>
<td>Peripheral Interface</td>
<td>UART/SDIO/SPI/I²C/I²S/IR Remote Control GPIO/ADC/PWM/LED Light &amp; Button</td>
</tr>
<tr>
<td>Operating Voltage</td>
<td>2.5 V 3.6 V</td>
</tr>
<tr>
<td>Network Protocols</td>
<td>IPv4, TCP/UDP/HTTP (HT20)</td>
</tr>
<tr>
<td>Frequency Range</td>
<td>2.4 GHz 2.5 GHz</td>
</tr>
<tr>
<td>User Configuration</td>
<td>AT Instruction Set, Cloud Server, Android/iOS App</td>
</tr>
</tbody>
</table>

3.9. Arduino Mega

Arduino Mega became popular with its famous series which is Arduino within the hobbyist after its release of Arduino Uno. These boards opened a new horizon for the embedded system community as it was easily configurable and programmable. Soon open source community created a wide variety of libraries and support electronic systems which made its support environment bigger. Due to the robustness, the people who are not in the embedded system community quickly embraced this system. The Arduino Mega is powered by the famous ATMega 2560 which is developed by Atmel which is now owned by Microchip. Atmel was a pioneer in the embedded system design for many years and the choice of this microprocessor is, without doubt, a great choice as it is very cheap and very powerful. The following shows
some capabilities of this system.

Table 2. Arduino Mega features [41, 42]

<table>
<thead>
<tr>
<th>Feature</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating Voltage</td>
<td>5V</td>
</tr>
<tr>
<td>Digital I/O Pins</td>
<td>54</td>
</tr>
<tr>
<td>Analog Input Pins</td>
<td>16</td>
</tr>
<tr>
<td>PWM Pins</td>
<td>15</td>
</tr>
<tr>
<td>Communication Protocols</td>
<td>SPI UART and I²C</td>
</tr>
<tr>
<td>Clock Speed</td>
<td>16 MHz</td>
</tr>
</tbody>
</table>

3.10. Texas Instruments TMS320 DSP on Spectrum Digitals eZdsp F28335

The Spectrum Digital eZdsp is a development environment that is made on top of the Texas Instruments TMS320 DSP (Digital Signal Processor). This processor is in the high end of the microcontrollers and has higher capabilities. This has a mid-level price range but has higher processing capabilities. Texas Instruments Code Composer environment is used to develop compile and write the code to the flash memory of the development environment.

Table 3. Texas Instruments TMS320F28335 features [43]

<table>
<thead>
<tr>
<th>Feature</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating Voltage</td>
<td>3.3V</td>
</tr>
<tr>
<td>Digital I/O Pins</td>
<td>64</td>
</tr>
<tr>
<td>Analog Input Pins</td>
<td>16</td>
</tr>
<tr>
<td>PWM Pins</td>
<td>18</td>
</tr>
<tr>
<td>Communication Protocols</td>
<td>SPI SCI CAN McBSP and I²C</td>
</tr>
<tr>
<td>Clock Speed</td>
<td>150 MHz</td>
</tr>
</tbody>
</table>

3.11. Real Time Clock (RTC) with BQ32002D

The real-time clock IC given has the primary work of providing time stamps to the requested device. This device has accuracy for the nearest second and cannot go beyond the seconds to calculate lower time scales. The data provided in the text is taken from the datasheet provided by Texas Instruments [44].

This device supports an 8-Pin SOIC pin configuration. At the use, we have used a pin expansion so that this is supported to through-hole application. This supports the VCC of 3.3V and is connected directly to the NodeMCU for its use. The oscillator used in the design is 32.768kHz. The specific reason for this odd number is that it can be easily divided to 1Hz in the application.
The circuit was implemented as given in the document with minor changes and has the following circuit diagram. The backup battery was not included in the circuit as we are not interested in getting the accurate time we keep 0 as the beginning of time and use the timestamps as increments of time as needed.

Two 47nF capacitors are used instead of 1µF capacitor and the interrupt pin of the chip is not used as we are not interested in the feedback within the register.

The Arduino wire library corresponds to 7 bit addressing the datasheet shows that to write to the IC we have to use 0XD0 address and 0XD1 as the write first 7 bits being the address and last bit is R/W. in Arduino the Wire.beginTransmission() will add 0 as the last bit and Wire.requestFrom() will add 1 as the last bit or the zeroth position.

Figure 10. Circuit diagram of BQ32002D cited from the data sheet and changed according to the need of the text [44]

The Figure 10 shows the circuit that should be setup in order to achieve communication and enable the functions of the BQ32002D.

The following figure which is Figure 11 shows the communication specification provided for the IC. Note that the read bit (R) is taken as 1 and write bit (W) is taken as 0 in this case.
Figure 11. Communication diagram of BQ32002D cited from the data sheet and changed according to the need of the text [44]

To write any data for the IC we use the following in NodeMCU,

```cpp
Wire.beginTransmission(CLOCK_ADDRESS);
Wire.write(REGISTER_ADDRESS);
Wire.write(DATA);
Wire.endTransmission();
```

To read any data for the IC we use the following in NodeMCU,

```cpp
Wire.requestFrom(CLOCK_ADDRESS, NUMBER_OF_BYTES_READ);
while(Wire.available()) {
    char c = Wire.read();
}
```

The NodeMCU uses D1 as the SCL and D2 as SDA these two are GPIO 5 and 4 respectively in case of the pin description using numbers.

The main concern is to start the device clock at 00:00:00 and increment from there to do this we have to access the device registers. The register access is straightforward according to the timing and communication diagram are given above.
Figure 12. Register diagram of BQ32002D cited from the data sheet and changed according to the need of the text [44]

The Figure 12 has the register structure and the register description of the IC is as given,

Table 4. Register details of the BQ32002 [44]

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Address</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SECONDS</td>
<td>0x00</td>
<td>0XXXXXXb</td>
<td>Clock seconds and STOP bit</td>
</tr>
<tr>
<td>MINUTES</td>
<td>0x01</td>
<td>1XXXXXXb</td>
<td>Clock minutes</td>
</tr>
<tr>
<td>CENT_HOURS</td>
<td>0x02</td>
<td>XXXXXXXXb</td>
<td>Clock hours, century, and CENT_EN bit</td>
</tr>
</tbody>
</table>

and the following gives the details of each registry component.

Table 5. Bit description of each register of the BQ32002 [44]

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STOP</td>
<td>Oscillator stop. The STOP bit is used to force the oscillator to stop oscillating. STOP is set to 0 on initial application of power, on all subsequent power cycles STOP remains unchanged. On initial power application STOP can be written to 1 and then written to 0 to force start the oscillator. 0 Normal 1 Stop</td>
</tr>
</tbody>
</table>
**10_SECOND**  
BCD of tens of seconds. The 10_SECOND bits are the BCD representation of the number of tens of seconds on the clock. Valid values are 0 to 5. If invalid data is written to 10_SECOND, the clock will update with invalid data in 10_SECOND until the counter rolls over; thereafter, the data in 10_SECOND is valid. Time keeping registers can take up to 1 second to update after the RTC switches from backup power supply to main power supply.

**1_SECOND**  
BCD of seconds. The 1_SECOND bits are the BCD representation of the number of seconds on the clock. Valid values are 0 to 9. If invalid data is written to 1_SECOND, the clock will update with invalid data in 1_SECOND until the counter rolls over; thereafter, the data in 1_SECOND is valid. Time keeping registers can take up to 1 second to update after the RTC switches from backup power supply to main power supply.

**OF**  
Oscillator fail flag. The OF bit is a latched flag indicating when the 32.768-kHz oscillator has dropped at least four consecutive pulses. The OF flag is always set on initial power-up, and it can be cleared through the serial interface. When OF is 0, no oscillator failure has been detected. When OF is 1, the oscillator fail detect circuit has detected at least four consecutive dropped pulses.

- 0  No failure detected
- 1  Failure detected

**10_MINUTE**  
BCD of tens of minutes. The 10_MINUTE bits are the BCD representation of the number of tens of minutes on the clock. Valid values are 0 to 5. If invalid data is written to 10_MINUTE, the clock will update with invalid data in 10_MINUTE until the counter rolls over; thereafter, the data in 10_MINUTE is valid. Time keeping registers can take up to 1 second to update after the RTC switches from backup power supply to main power supply.

**1_MINUTE**  
BCD of minutes. The 1_MINUTE bits are the BCD representation of the number of minutes on the clock. Valid values are 0 to 9. If invalid data is written to 1_MINUTE, the clock will update with invalid data in 1_MINUTE until the counter rolls over; thereafter, the data in 1_MINUTE is valid. Time keeping registers can take up to 1 second to update after the RTC switches from backup power supply to main power supply.
CENT.EN Century enable. The CENT.EN bit enables the century timekeeping feature. If CENT.EN is set to 1, then the clock tracks the century using the CENT bit. If CENT.EN is set to 0, the clock ignores the CENT bit.

0 Century disabled
1 Century enabled

CENT Century. The CENT bit tracks the century when century timekeeping is enabled. The clock toggles the CENT bit when the year count rolls from 99 to 00. Because the clock compliments the CENT bit, the user can define the meaning of CENT (1 for current century and 0 for next century, or 0 for current century and 1 for next century).

10.HOUR BCD of tens of hours (24-hour format). The 10.HOUR bits are the BCD representation of the number of tens of hours on the clock, in 24-hour format. Valid values are 0 to 2. If invalid data is written to 10.HOUR, the clock will update with invalid data in 10.HOUR until the counter rolls over; thereafter, the data in 10.HOUR is valid. Time keeping registers can take up to 1 second to update after the RTC switches from backup power supply to main power supply.

1.HOUR BCD of hours (24-hour format). The 1.HOUR bits are the BCD representation of the number of hours on the clock, in 24-hour format. Valid values are 0 to 9. If invalid data is written to 1.HOUR, the clock will update with invalid data in 1.HOUR until the counter rolls over; thereafter, the data in 1.HOUR is valid. Time keeping registers can take up to 1 second to update after the RTC switches from backup power supply to main power supply.

3.12. Analog to Digital Converter with ADS7828

The ADS7828 is a widely used ADC among hobbyist and Arduino Uno users to expand the capabilities in the Uno. The ADC converter is said to be working in the 2.5V and has an internal reference to 2.5V. But the device can also be used in 5V with a 5V reference to the Ref pin. The device supports fast I²C communication in 400kHz. There are two devices used in the system so the address pins A₀ A₁ can be used to distinctly separate the addressees of the two devices. There are some established libraries for the device but we are going to create our library based on the best practices by all of the other libraries to achieve efficiency and to remove redundant operations. As an example, the library ADS7828 has a voltage conversion written inside the library which we are not going to use. Instead, we are going to get the raw values and pass them to transmission. The standard conversion is given below.
If the total voltage range is $V_R$ and the received value is $t$ the highest value of 12 bit number system is 4096 Then the real voltage value or $V$ is,

\[
V = \frac{(V_R \times t)}{4096}
\]  

(1)

Figure 13. The ADC IC circuit diagram cited from data sheet and modified for the purpose of this text [45]

The address for the device is as shown with $A_1$ and $A_0$ can be configured by changing the electrical signals sent to the pins as given in Figure 13. after sending the address byte a command byte is sent to read the values in the registers. The result of a reading is given in two bytes where first byte contains D11 to D8 and next byte contains the rest. The Figure 14 give a summery of the ADC IC’s data sheet.
3.13. 4-Bit 1-of-2 FET Multiplexer/Demultiplexer with SN74CBT3257C

This IC is used to change the input and output lines connected. This acts as a simple switch that switches between two inputs to connect them to one output. The 4 bit 1 of 2 is used as when the number of pins increases the package size becomes more small and unmanageable. The IC’s major usage is switching and the use of the FET’s (Field Effect Transistors) is another basic quality that was required for the design as this multiplexing is done through PWM and ADC the need to pass analog signals through the IC without dropping or changing their voltage. The undershoot is the phenomenon when the output voltage is lower than the desired voltage. The Figure 15 has the summery of the Multiplexer IC’s data sheet.
Figure 15. Condensed data on the SN74CBT3257C [46]
4. Approach

4.1. Device Design

The device we are proposing has several modular parts. The figure ?? shows the high-level block diagram of the design. Also, some of the naming conventions are clarified as below. Note that the simulator device can be turned into a monitoring device so the device we are proposing is fulfilling two major roles one being a simulator and another role will be as a monitoring device.

![High level block diagram of the system](image)

1. Data gathering PC - This PC is responsible for issuing commands to the device and gathering information inside the device and storing them.

2. HIL device - This total electrical system is the device which we are proposing that is connected to the embedded system that is tested.

3. WiFi module - The device which communicate with the PC and get all information to the HIL device.
4. Peripherals - These are optional devices in some of the designs which include ex-ternal ADC and Multiplexer switch (The Multiplexer switch is always connected to the simulator and is in between the states of peripheral device and simulator in its actions).

5. Simulator module - This device is responsible to simulate the environment to target embedded device.

6. Host device - This is the targeted embedded system which is tested.

7. Sensors and Actuators - This is optional but if some person needs to record the data that is going through the embedded system when it is connected to the environment then the devices in the actual environment which are sensors and actuators can be connected to the HIL module this can be used for debugging purposes of the sensors and actuators.

As mentioned above the system should act as a simulator and monitor so to achieve this target a special switching mechanism using a multiplexer is used the following diagram figure ?? shows the high-level view of that design.

![Figure 17. High level block diagram of the HIL module data bus](image)
WiFi Module

The WiFi module is responsible to keep the connection between the computer and the HIL system which makes it the most vital component in the system. The figure ?? shows the proposed system. The system also handles two communication protocols at the same time one between pc and itself and the other between itself and the HIL system. The WiFi system also controls the Timing of the system such as if the PC needs time stamps they are provided by this module itself. The timing feature can be activated or deactivated according to the user’s wish.

![Circuit diagram of the WiFi module](image)

By looking at the above data on the RTC IC it is concluded that the algorithm for the data that is received from the IC should be designed in the following form to create the time stamp inside the NodeMCU.

```
String timestamp = "";
int i = 0;
while (Wire.available()) // responder may send less than requested
{
    char c = Wire.read(); // receive a byte as character
```
int p = c & B00001111;
int q = (c >> 4) & B00000111;
timestamp += BCDtoString(q, DEC);
timestamp += BCDtoString(p, DEC);
if (i != 2)
    timestamp += ":";
i++;
}

Analog Reader

The Analog reader module is specifically designed for the reading of 16 analog inputs. The basic necessity for this design is to compare the analog converters built into the Arduino and TMS320 and compare them to get the best performing device to the final design of the system. The analog reader module as described will be housing for 16 analog inputs while communicating with the main simulator in I²C communication. The figure ?? shows the proposed system.

![Circuit diagram of the analog reader](image)

**Figure 19. Circuit diagram of the analog reader**
A code snippet to harvest the data on channel 0 is given below.

```cpp
Wire.begin();
unsigned int value = 0;
Wire.beginTransmission(0x48);
Wire.write(0x8F); //Command with PD1, PD0, SD = 1
Wire.endTransmission();
Wire.requestFrom(0x48, 2);
while (Wire.available()) {
    value = value << 8;
    value |= Wire.read();
}
```

**Multiplexer Switch**

The multiplexer switch shown in figure ?? is very important to keep the two functions of the system which are monitoring and simulating without conflicting with each other. The system is isolating the output by the simulator from the sensor input and actuator output. This is done through an array of the FET switches so the voltage is not misinterpreted between the gates of the switches. The lines for the PWM from the device should be separately handled as they do not have any HIL simulator pins attached to them.
Device 1: Module with Arduino Mega Simulator and Peripherals

The HIL simulator is made up of Arduino Mega, ADC, and Multiplexer switching in this setup and shown in figure ???. The performance of this device is monitored using the light operating example given in the document and analyzed. The device always should be included with the switching mechanism but a ADC module is added for the device this is done to make sure that the processing is quick enough to outperform the internal ADC mechanisms.
Device 2: Module with Arduino Mega Simulator only

The HIL simulator is made up of Arduino Mega and Multiplexer switching in this setup and shown in figure ???. This system uses the ADC module built-in inside the Arduino Mega. The performance of this device is monitored using the light operating example given in the document and analyzed. The systems own ADC is used to determine the efficiency of the internal operations.

Device 3: Module with TMS320 DSP Simulator and Peripherals

The HIL simulator is made up of TMS320 DSP, ADC, and Multiplexer switching in this setup and shown in figure ???. The performance of this device is monitored using the light operating example given in the document and analyzed. The device always should be included with
the switching mechanism but a ADC module is added for the device this is done to make sure that the processing is quick enough to outperform the internal ADC mechanisms.

Figure 23. Device 3 Block Diagram

Device 4: Module with TMS320 DSP Simulator only

The HIL simulator is made up of TMS320 DSP and Multiplexer switching in this setup and shown in figure ???. This system uses the ADC module built-in inside the TMS320 DSP. The performance of this device is monitored using the light operating example given in the document and analyzed. The systems own ADC is used to determine the efficiency of the internal operations.

Figure 24. Device 4 Block Diagram
4.2. Communication Design

The communications in this system are done in bytes. This makes the smallest unit a byte and this is used to build a packet and a frame in the communication. The message protocols have variable lengths but the start of 3 bytes of the message will determine the length of the message. The Packet is started by 0x01 and ended by 0x04 and Frames start is 0x02 and the end is 0x03. The total structure of a message is shown below in figure ??.

![Packet and Frame Diagram]

Figure 25. Example of a message in the system

The communication and the devices changes can be taken as a state machine. This is because most of the embedded systems and their process can be shown as finite state machines.

**Between WiFi Module and Simulator**

The communication between the WiFi module and Simulator is done through the UART communication. In this design, we have to convert the signal from 3.3V to 5V and vice versa for communicating with boards related to node MCU and the main controller. To achieve this task we need a level converter in the UART communication bus. Also, a null modem connection has been established wherein one device RX is connected to other devices TX for the communication this can be shown in figure below.

![State Diagram]

Figure 26. State diagram of the WiFi system in communication
Table 6. State diagram description

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Recive Poll or Idle or Communicating with Computer</td>
</tr>
<tr>
<td>B</td>
<td>Simulation</td>
</tr>
<tr>
<td>C</td>
<td>Monitoring</td>
</tr>
<tr>
<td>a</td>
<td>Not Ready</td>
</tr>
<tr>
<td>b</td>
<td>Ready to Communication (Completed communication with Computer)</td>
</tr>
<tr>
<td>c</td>
<td>Sending Data To Simulator</td>
</tr>
<tr>
<td>d</td>
<td>Done sending simulator Data</td>
</tr>
<tr>
<td>e</td>
<td>Get Samples (no of samples reduced)</td>
</tr>
<tr>
<td>f</td>
<td>Samples Done</td>
</tr>
</tbody>
</table>

The following is the Data Frame details of the communication. If the packet is sent to the WiFi module it is shown as N and sending to simulator is shown as S in the direction.

Table 7. Poll data frame direction (N)

<table>
<thead>
<tr>
<th>Position</th>
<th>Size (in bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>1</td>
<td>Has Hex value 0x05</td>
</tr>
</tbody>
</table>

Table 8. Ideal data frame direction (S)

<table>
<thead>
<tr>
<th>Position</th>
<th>Size (in bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>1</td>
<td>Has Hex value 0x06</td>
</tr>
</tbody>
</table>

Table 9. Start data frame direction (S)

<table>
<thead>
<tr>
<th>Position</th>
<th>Size (in bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>1</td>
<td>Has Hex value 0x0A</td>
</tr>
</tbody>
</table>

Table 10. Operation data frame direction (S)

<table>
<thead>
<tr>
<th>Position</th>
<th>Size (in bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>1</td>
<td>Has Hex value 0x07 for simulation and 0x09 for monitoring</td>
</tr>
</tbody>
</table>
Table 11. Simulator data frame direction (S)

<table>
<thead>
<tr>
<th>Position</th>
<th>Size (in bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>1</td>
<td>Has Hex value 0x08</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>GPIO value</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>PWM value</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>Time</td>
</tr>
</tbody>
</table>

Table 12. Monitoring data frame direction (N)

<table>
<thead>
<tr>
<th>Position</th>
<th>Size (in bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>1</td>
<td>Has Hex value 0x0B</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>GPIO value</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>ADC value</td>
</tr>
</tbody>
</table>

Between WiFi Module and Computer

The WiFi module and computer speaks through the wifi network supported to them. This module cannot access 5G bandwidth so we are using normal 2.4 GHz bandwidth. The associated state machine is shown in figure below.

![State diagram of the WiFi system in communication](image)

Figure 27. State diagram of the WiFi system in communication
Table 13. State diagram description

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Node Idle</td>
</tr>
<tr>
<td>B</td>
<td>Node Ready Communicate</td>
</tr>
<tr>
<td>C</td>
<td>Simulation</td>
</tr>
<tr>
<td>D</td>
<td>Monitoring</td>
</tr>
<tr>
<td>a</td>
<td>Check server</td>
</tr>
<tr>
<td>b</td>
<td>Communication initialization</td>
</tr>
<tr>
<td>c</td>
<td>Simulator</td>
</tr>
<tr>
<td>d</td>
<td>Monitoring</td>
</tr>
<tr>
<td>e</td>
<td>Get Samples (no of samples reduced)</td>
</tr>
<tr>
<td>f</td>
<td>Samples Done</td>
</tr>
<tr>
<td>g</td>
<td>Send Samples</td>
</tr>
<tr>
<td>h</td>
<td>Samples Done Communication Wrap Up</td>
</tr>
</tbody>
</table>

The following is the data frame details of the communication. If the packet is sent to the WiFi module it is shown as N and sending to computer is shown as C in the direction.

Table 14. Poll data frame direction (C)

<table>
<thead>
<tr>
<th>Position</th>
<th>Size (in bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>1</td>
<td>Has Hex value 0x08</td>
</tr>
</tbody>
</table>

Table 15. Start data frame direction (N)

<table>
<thead>
<tr>
<th>Position</th>
<th>Size (in bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>1</td>
<td>Has Hex value 0x06</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>The sample size</td>
</tr>
</tbody>
</table>

Table 16. Stop data frame direction (C)

<table>
<thead>
<tr>
<th>Position</th>
<th>Size (in bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>1</td>
<td>Has Hex value 0x0E</td>
</tr>
</tbody>
</table>
Table 17. Time data frame direction (C)

<table>
<thead>
<tr>
<th>Position</th>
<th>Size (in bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>1</td>
<td>Has Hex value 0x0A</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>Time value</td>
</tr>
</tbody>
</table>

Table 18. Simulator data frame direction (N)

<table>
<thead>
<tr>
<th>Position</th>
<th>Size (in bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>1</td>
<td>Has Hex value 0x05</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>GPIO Pin Configuration and Value</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>PWM values</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>Time</td>
</tr>
</tbody>
</table>

Table 19. Monitoring data frame direction (C)

<table>
<thead>
<tr>
<th>Position</th>
<th>Size (in bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>1</td>
<td>Has Hex value 0x07</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>GPIO value</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>ADC value</td>
</tr>
</tbody>
</table>

4.3. Playbook Implementation

The playbook has the simulation techniques after sending this data to the simulator it will play all the steps while recording the data. A time is allocated to each play in the playbook this time is taken as the delay between two plays.

4.4. Software Design

The Server in the computer is written in Python and can connect at any time. The server mostly uses text files as inputs and output text files in csv format which can be used for later processing using a common office or other packages.

4.5. Cost for Devices

The cost of each of the individual devices is given below so the raw cost of the electrical components needed for each device can be derived. Note that cost for components like resistors, capacitors, or oscillators is not given also price of TMS320 development board is given by looking at second-hand value as they are no longer sold.
Table 20. Cost for each devices major electrical components

<table>
<thead>
<tr>
<th>Device</th>
<th>Components</th>
<th>Cost (Aprox.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Arduino Mega, ADC, NodeMCU, Timer, Switch</td>
<td>50 Dollars</td>
</tr>
<tr>
<td>2</td>
<td>Arduino Mega, NodeMCU, Timer, Switch</td>
<td>40 Dollars</td>
</tr>
<tr>
<td>3</td>
<td>TMS320, ADC, NodeMCU, Timer, Switch</td>
<td>250 Dollars</td>
</tr>
<tr>
<td>4</td>
<td>TMS320, NodeMCU, Timer, Switch</td>
<td>250 Dollars</td>
</tr>
</tbody>
</table>

4.6. Example Setup

The first experiment that we are setting up is to decide the speed of the GPIO data retrieval. In this setup, we will create a unique pattern of digital inputs and we will identify how much time it will consume by each system. Then a unique digital output is given to be generated to the system and will analyze the time taken for the process.

The second experiment is on the ADC’s performance and they are monitored in two patterns. In this setup, we will create 16 inputs of analog signals which have 8 voltage divider circuits and 8 PWM input. In the experiment, we will consider the values and their error rate with the time consumed for each module in the process.

The third test comprises generating PWM signals and testing their time to do so in the process. The error rates of the generated signals are monitored in the process.

The fourth test is to measure the time taken to create a timestamp in the WiFi module is done. This also includes analysis of the data transfer rate between the simulator and device and PC with the device through WiFi.

The common test setup is a toy experiment which is a light dimming circuit the PWM signal is connected to a LED with a button to the system. The system works as follows. The button should be pressed to turn on the light bulb then the trimmer should be turned to change the brightness of the LED. If the button is pressed again the bulb will be turned off. The code that is used for the example is given below. The figure ?? shows the circuit diagram.
Figure 28. Circuit diagram of the test system

```c
#include <Arduino.h>

int set = 0;

void setup() {
  Serial.begin(9600);
  pinMode(3, OUTPUT);
  pinMode(2, INPUT);
}

void loop() {
  if (set == 1) {
    int val = analogRead(A0);
    analogWrite(3, val / 4);
  } else {
    analogWrite(3, 0);
  }
  if (digitalRead(2) == LOW) {
    if (set == 1)
      set = 0;
    else
      set = 1;
    delay(1000); // simple debounce
  }
}
```
5. Data Explained

We have taken all the samples and calculated the standard error as follows,

\[ s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})}{(n - 1)}} \]  

(2)

\[ \text{error} = \frac{s}{\sqrt{n}} \]  

(3)

where,
\( \bar{x} \) - mean of sample
\( s \) - standard deviation
\( n \) - number of samples

5.1. Performance of GPIO pins

We have setup the classical binary counter on an 8 LED strip. The setup is as shown in the figure and this is testing the time to apply a GPIO output to the system. In Arduino Mega port A is pin 22 to 29 these pins are used for the testing.

We have collected 10 different sets of 300 samples of output from the port of Arduino Mega (to include all 255 combinations and extra values) which result in,

<table>
<thead>
<tr>
<th>Sample Set (Each have 300 samples)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average (µs)</td>
<td>3.2</td>
<td>3.2</td>
<td>3.2</td>
<td>3.2</td>
<td>3.2</td>
<td>3.2</td>
<td>3.2</td>
<td>3.2</td>
<td>3.2</td>
<td>3.2</td>
</tr>
<tr>
<td>SD (µs)</td>
<td>1.7</td>
<td>1.7</td>
<td>1.7</td>
<td>1.7</td>
<td>1.7</td>
<td>1.7</td>
<td>1.7</td>
<td>1.7</td>
<td>1.7</td>
<td>1.7</td>
</tr>
<tr>
<td>Error (µs)</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

This results in for 3.2 ± 0.1 µs to output 8-bit value.

When testing the input of Arduino Mega we have connected the GPIO pins to a configuration to give 10101010 8-bit value here we also calculate the deviation of the reading with time.

The readings had not diverted from the original values. The time to read is as given in the table for 10 different sets of 300 samples of output for the port.
Table 22. Arduino port input time

<table>
<thead>
<tr>
<th>Sample</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average (µs)</td>
<td>3.0</td>
<td>3.0</td>
<td>3.0</td>
<td>3.0</td>
<td>3.0</td>
<td>3.0</td>
<td>3.0</td>
<td>3.0</td>
<td>3.0</td>
<td>3.0</td>
</tr>
<tr>
<td>SD (µs)</td>
<td>1.7</td>
<td>1.7</td>
<td>1.7</td>
<td>1.7</td>
<td>1.7</td>
<td>1.7</td>
<td>1.7</td>
<td>1.7</td>
<td>1.7</td>
<td>1.7</td>
</tr>
<tr>
<td>Error (µs)</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

This results in for $3.0 \pm 0.1 \, \mu s$ to get input of 8-bit value. When we do the same test for the TMS 320 we are reading a 32 Data register to get the values this resulted 36 cycles in 150MHz both input and output tests which lead to 0.24 $\mu s$ for a read or write.

5.2. Performance of ADC in Arduino, TMS320 and Peripheral Device

We have made 8 voltage dividers and 8 PWM generators and have given 16 inputs to each testing platform while taking 100 samples. The sample values and time to read are recorded and diversion from original values is recorded. In TMS320 only the voltages from 0 to 3V are taken so we have made some values within the range of 0 to 3V and others over that range.

This results in a value variation of 0.05V in real analog input in $1793.6 \pm 10.10 \, \mu s$. And for PWM inputs there should be a low pass filter to be between the ADC and PWM as shown in figure ?? and it will be stable in 44920 $\mu s$ with $\pm 0.01V$ accuracy.

The Arduino with ADC peripheral device for 100 samples has 0.5V variation with time $8826.9 \pm 5.90 \, \mu s$. This does not have a delay in low pass filter and shares the same accuracy of normal analog reading in device with 0.5V variation.

In TMS320 built-in ADC has 0.01V variations and can detect data in $0.91 \, \mu s$. And for PWM inputs there should be a low pass filter to be between the ADC and PWM and it will be stable $1.8 \, \mu s$ with $\pm 0.1V$ accuracy.

The TMS320 with ADC peripheral device for 100 samples has 0.5V variation with time $402.9 \pm 0.09 \, \mu s$. This does not have a delay in low pass filter and shares the same accuracy of normal analog reading in device with 0.5V variation.

![Figure 29. Circuit diagram of the PWM to ADC filter](image-url)
The ADC data comparison is shown as in the figure ???. The data shows that the ADC component attached has high latency and higher error rate than the inbuilt ADC components in Arduino and TMS320.

5.3. Performance of PWM

The values of the PWM signals have ± 0.01V accuracy in Arduino as given above but takes 44920 µs to stabilize to that value. By taking 100 samples we tried to find the time to generate PWM signals which in return gave us 75.88 ± 1.91 µs.

The values of the PWM signals have ± 0.1V accuracy in TMS320 as given above but takes 1.8 µs to stabilize to that value. By taking 100 samples we tried to find the time to generate PWM signals which in return gave us 2.38 µs.

5.4. Performance of WiFi Module

In the WiFi module we did not test the network connection or data transfer rates as it is depending on the network infrastructure and the data that is send at the given time. But as there is a computation or acquiring of time from a RTC we have tested the average time that is taken to acquire time from RTC and it is 439.8 ± 35.94 µs.
5.5. Sample setup

As from the above test data we see that the accuracy in the Arduino is nearly as good as TSM320 and the data acquiring speed is near to TSM 320 in µs level. By looking at the costs of the devices we can conclude that we can use Device 2 which is the device which uses the Arduino Mega as the simulator only as the model device for this experiment. By using the example setup we have seen that the function of simulating and monitoring works properly for 4 samples/second speed.
6. Further Work

The System can implement a method to capture the communications that are going through the embedded systems. Many systems mainly depend on the communication systems that are established in them. In this project, although we used communications such as **I^2C** and UART we have not integrated them into the HIL system. The speed of the **I^2C**, UART, and SPI is high so the GPIO is unable to tap into their communications. The speed of the detection is vital in this process and the devices with such speed are costly. The main reason for not using the existing modules in the simulators is that some communications cannot have two or more controllers in the communications and some devices have made the digital designs attached to the device address so spoofing into other communications is blocked.

Another main aspect that we are looking into in the project is that make separate systems to create PWM without using the internal system to do that process this makes the generation of PWM independent with high resolutions. Another main part is using a separate GPIO component as well. By combining all of them we are going to remove the simulator module (Arduino or TSM320 DSP) and make an independent system that is purely electrically designed and connected to the WiFi module only for communication. This will help to work with any type of voltage levels like 5V, 3.3V, or 1.8V of the embedded systems.

The simulation software can be further improved to visualize the performance better. Also, the playbook writing can be improved using this mechanism. Eventually, replacing the PC and establishing a cloud-based implementation is another approach that is under consideration for this design.
7. Conclusion

By referring to the above details it is obvious that the HIL simulation is a very important aspect of embedded system design. The use and development around these systems are very little due to the issues in systems cost and complexity. But this system is very valuable for small designers and students when they are developing embedded systems.

The HIL systems can be taken as a method of black-box testing and white box testing for the developers. If this system is inexpensive considerable amount of resources equipment and time can be saved. The HIL systems as shown by the related work is widely used to save time and resources. This system is used to test and develop embedded systems to critical and expensive systems like vehicles where destructive testing becomes a devastating approach.

Many of the systems that are made currently are proprietary and domain specific. These systems also cost millions and need special expertise for operation. The user groups like individuals, electronic enthusiast, hobbyist and students cannot afford such systems. But this group also burn a lot of devices while they are developing embedded systems plus they spend much time on figuring out the errors in the system without any indication of where error is due to the lack of proper debugging tools. So we have thought of designing a system which can monitor and simulate embedded system but for a very lower cost for non-real time systems.

We have selected the Arduino, Node MCU and TSM320 as our base electrical devices that is helping to create a HIL. we have created 4 devices which are tested under same conditions to be cost effective, accurate and, efficient. The system is consist of a simulation element a basic software to record the data and a WiFi communication module to communicate data between the simulator and the software.

In this project, we also identified that NodeMCU has flaws in its design as an I²C responder device and to be functioning as an SPI device. This also opened a door to look into some of the flaws in NodeMCU based devices that are not visible to the end-users at the first glance. This is because the NodeMCU played a major role in WiFi communication.

The test results shown above gives that Device 2 (Arduino Mega and switching module only the outside ADC module is not used) Design is more cost-effective and efficient on simulation related to slowly functioning embedded systems as its accuracy and detect-ability speed is higher than devices using the peripheral devices. This means that this system can be used to test non-real time application based embedded systems. This also prevent of the users burning up electrical elements due to minor errors in the algorithms of the system that they are designing.
References


[23] Zhiwang Feng, Rafael Pena-Alzola, Paschalis Seisopoulos, Efren Guillo-Sansano, Mazheruddin Syed, Patrick Norman, and Graeme Burt. A scheme to improve the


[33] ElectronicsTutorials. The demultiplexer.

[34] ELPROCUS. What is analog to digital converter & its working.

[35] JIMBLOM. Serial communication.

[36] SFUPTOWNMAKER by sparkfun. I2c.

[37] SFUPTOWNMAKER by sparkfun. I2c.

[38] SFUPTOWNMAKER by sparkfun. I2c.


[41] Microchip. *ATmega640/4-1280/4-1281/4-2400/4-2560/4-2561/V*.


Vita

Chanuka Sandaru Elvitigala was born in Colombo, Sri Lanka in 1993 to Gamini Elvitigala and Hashini Elvitigala. He completed his primary and secondary education in Isipathana College, Colombo 5. He studied in English medium until the GCE O/L then studied in Physical Stream in his GCE A/L and held the best results award in Physical Stream 2012 A/L at Isipathana College. He is also an all-island western music award holder for 3 consecutive years. He joined the Faculty of Information Technology of the University of Moratuwa, Katubedda, Sri Lanka to pursue B.Sc in Information Technology (Hons.) in English medium. He graduated in 2018 with a first-class. While he was attending the university he participated in the Japanese Robotics and AI student Exchange program two times one in Sri Lanka and the other in Japan. He also published 5 international publications in IEEE under the topic, “Adaptive navigation and motion planning for a mobile track robot.” IFSA-SCIS 2017 (Japan), “Machine Learning Capable, IoT Air Pollution Monitoring System with Upgradable Sensor Array.” ISIS 2017 (South Korea), “IoT Proactive Disaster Management System for Mines.” ISIS 2017 (South Korea), “An Ad-Hoc Network based on Low-cost Wi-Fi Device for IoT Device Communication.” ICITR 2017 (Sri Lanka) and “Low Cost and IoT based Greenhouse with Climate Monitoring and Controlling System for Tropical Countries.” ICSSE 2018 (Taiwan).

He also worked in various companies, Durdans Hospitals (As a part-time and time to time a full time) Software Engineer 2015 - 2018, Virtusa Pvt Ltd Software Engineer 2018 - 2019, TRUCE Software Software Engineer 2019 - 2019.

He moved to the USA in 2019 and settled in Baton Rouge. He started working on TRUCE Software in September 2019 and then resigned in December to Join LSU. In 2020 Spring he joined the Department of Electrical and Computer Engineering at Louisiana State University to pursue his Master of Science in Computer Science. At present, he is working with Dr. Baumgartner. His hobbies are listening to music, playing games, and working on embedded systems as a hobbyist. He is planning to receive his Masters in May 2021.