

# **INTERNSHIP REPORT**

**On**

## **Arduino Board programming using MATLAB/Simulink**

A report submitted in partial fulfillment of the requirements for the Award of Degree of

<b>STUDENT NAME</b>	<b>CLASS</b>	<b>BRANCH</b>	<b>COLLEGE</b>
Arzoo Goyal	B. Tech (3 <sup>rd</sup> Yr)	Electronics & Comm	RBSETC, Agra
Devansh Yadav	B. Tech. (3 <sup>rd</sup> Yr)	Electronics & Comm	RBSETC, Agra
Payal Mohnani	B. Tech. (3 <sup>rd</sup> Yr)	Electronics & Comm	RBSETC, Agra

## **Training Period**

(June 30, 2023- Aug 17, 2023)

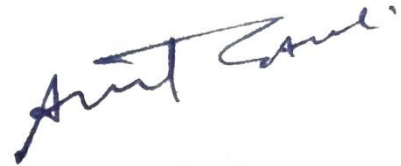
## **Under the Guidance of**

Sh. AJEET GAUR  
Scientist 'E'

**Aerial Delivery Research & Development Establishment  
Defense Research & Development Organization  
Ministry of Defense, Government of India  
Agra Cantt – 282001**

## CERTIFICATE

This is to certify that the project report compiled by **Ms. Arzoo Goyal, Mr. Devansh Yadav & Ms. Payal Mohnani** which is entitled "Arduino Board programming using Matlab/Simulink" is an authentic record of the effort carried out by them during the period of summer training from 30 June 2023 to 17 August 2023. The report is aimed towards the partial fulfillment of the requirement of the training certificate duly accredited by Aerial Delivery Research & Development Establishment (ADRDE), Agra under the guidance of **Mr. Ajeet Gaur, Scientist 'E'**.

A handwritten signature in blue ink, appearing to read 'Ajeet Gaur', is written diagonally across the page.

**Mr. Ajeet Gaur, Scientist 'E'**

**Project Guide**

## **Declaration**

We hereby declare that the work reported in the Aerial Delivery Research & Development Establishment (ADRDE) project entitled as "Arduino Board programming using Matlab/Simulink" In partial fulfillment for the award of certificate of DRDO submitted at Aerial Delivery Research & Development Establishment (ADRDE), Agra comprises only our original work done by us under the guidance of Mr. Ajeet Gaur, Scientist 'E', ADRDE, Agra during 30<sup>th</sup> June 2023 to 17<sup>th</sup> August 2023.

**Date: 17<sup>th</sup> August, 2023**

**Name of Students**

Arzoo Goyal

Devansh Yadav

Payal Mohnani

## **Acknowledgment**

Aerial Delivery Research & Development Establishment (ADRDE) is one of the premier establishments of Defense Research and Development organization (DRDO). This establishment is committed to provide the user's state-of-the-art product & services to its customers in areas of the aerodynamic decelerators, pneumatic structures, aircraft arrester barriers and allied systems through research and development, innovation, teamwork and following up the same by continual improvement based on user's perception.

I am highly obliged to Dr Manoj Kumar, Director ADRDE, Agra for allowing me to associate with this esteemed establishment as a summer trainee.

I express my sincere thanks to Mr. Ajeet Gaur, Scientist 'E' for his unflagging guidance throughout the progress of this project as well as valuable contribution in the preparation and compilation of the text.

I am thankful to all those who have helped me for the successful completion of my training at ADRDE Agra.

**Ms. Arzoo Goyal**

Branch: Electronics & Communication

B.Tech.: III Year

RBS, Agra

## **Acknowledgment**

Aerial Delivery Research & Development Establishment (ADRDE) is one of the premier establishments of Defense Research and Development organization (DRDO). This establishment is committed to provide the user's state-of-the-art product & services to its customers in areas of the aerodynamic decelerators, pneumatic structures, aircraft arrester barriers and allied systems through research and development, innovation, teamwork and following up the same by continual improvement based on user's perception.

I am highly obliged to Dr Manoj Kumar, Director ADRDE, Agra for allowing me to associate with this esteemed establishment as a summer trainee.

I express my sincere thanks to Mr. Ajeet Gaur, Scientist 'E' for his unflagging guidance throughout the progress of this project as well as valuable contribution in the preparation and compilation of the text.

I am thankful to all those who have helped me for the successful completion of my training at ADRDE Agra.

**Mr. Devansh Yadav**

Branch: Electronics & Communication

B.Tech.: III Year

RBS, Agra

## **Acknowledgment**

Aerial Delivery Research & Development Establishment (ADRDE) is one of the premier establishments of Defense Research and Development organization (DRDO). This establishment is committed to provide the user's state-of-the-art product & services to its customers in areas of the aerodynamic decelerators, pneumatic structures, aircraft arrester barriers and allied systems through research and development, innovation, teamwork and following up the same by continual improvement based on user's perception.

I am highly obliged to Dr Manoj Kumar, Director ADRDE, Agra for allowing me to associate with this esteemed establishment as a summer trainee.

I express my sincere thanks to Mr. Ajeet Gaur, Scientist 'E' for his unflagging guidance throughout the progress of this project as well as valuable contribution in the preparation and compilation of the text.

I am thankful to all those who have helped me for the successful completion of my training at ADRDE Agra.

**Ms. Payal Mohnani**

Branch: Electronics & Communication

B.Tech.: III Year

RBS, Agra

## **Abstract**

This report offers an in-depth exploration of the integration of Arduino with MATLAB and Simulink, showcasing their collaborative potential in constructing and evaluating real-world systems. The focus lies on the synthesis of Simulink models, subsequently deployed and tested on Arduino microcontroller boards.

Commencing with an overview of the range of available Arduino Single Board Computers (SBCs), a comprehensive comparative analysis is provided. Factors such as microcontroller specifications, power consumption, I/O configurations, memory capacity, communication protocols, embedded sensors, physical dimensions, weight, and cost-effectiveness are systematically evaluated.

The report then delves into the theoretical underpinnings of Simulink, elucidating its configuration for seamless interfacing with Arduino. A focal point is the development of tailored Simulink blocks for parsing GPS data, coupled with insights into establishing steadfast connections through Arduino's Hardware serial port.

In summation, this report underscores the synergy attainable by uniting Arduino, MATLAB, and Simulink, fostering a dynamic framework conducive to real-time control and simulation applications. The successful amalgamation of these tools offers a promising avenue for future advancements in the realms of control systems and simulation methodologies.

## List of Tables & Figures

Figure 1: Table of Boards from Nano Family .....	12
Figure 2: Table of Boards from MKR Family .....	13
Figure 3: Table of Boards from Classic Family .....	13
Figure 4: Install Support Package.....	15
Figure 5: Hardware Set-Up .....	16
Figure 6: Simulink Hardware Support Package.....	17
Figure 7: Block sub-libraries in Hardware Support Library .....	17
Figure 8: Run Model on Target Hardware .....	18
Figure 9: Class Definition in Matlab System Device Driver Block .....	19
Figure 10: Constructor Method .....	20
Figure 11: I/O and Termination Methods.....	20
Figure 12: Setting Up I/O Signal Properties .....	21
Figure 13: Building Artifacts.....	22
Figure 14: Adding Description for Users .....	22
Figure 15: Model for PWM signal Generation .....	24
Figure 16: Running Model PWM signal Generation Model on Hardware.....	24
Figure 17: Model for Sine Wave PWM Generation .....	25
Figure 18: Running Sine PWM Model on Hardware .....	25
Figure 19: Generation of Model for Half Sine Wave Signal .....	26
Figure 20: Running the Model on Hardware for Half Sine Wave Signal .....	26
Figure 21: GPGGA String Example .....	28
Figure 22: List of the data values given by GPGGA NMEA sentence (RF Wireless World, 2023). .....	28
Figure 23: Flow Chart of the GPS Parser program .....	30
Figure 24: Snapshot of Matlab Code for Driver Block .....	31
Figure 25: Simulink Block Code made using Driver Block.....	31
Figure 26: Testing of Code Block.....	32
Figure 27: Hardware Configuration for the Program .....	32



# Table of Content

1 Introduction .....	11
2 Literature Review .....	12
2.1 Literature Survey of various Arduino SBCs .....	12
3 Simulink.....	15
3.1 Simulink Support Package for Arduino Hardware .....	15
3.1.1 Setup and Configuration.....	15
3.1.1.1 Install Support Package.....	15
3.1.1.2 Hardware Setup .....	16
3.1.1.3 Open Block Library for Arduino Hardware .....	16
3.1.2 Modeling.....	17
3.1.3 Run on Target Hardware .....	17
Stop or Restart a Model on the Arduino Hardware .....	18
3.1.4 Device Driver Blocks .....	18
3.1.4.1 Class Definition .....	19
Constructor Method .....	19
Initialization, Output and Termination Methods.....	20
Input and Output Signal Properties .....	20
Build Artifacts.....	21
Write the Hardware-specific C/C++ Code.....	22
Description for Users .....	22
4 Simulation .....	23
4.1 Generation of various signals using Simulink and Arduino Hardware:.....	23
4.1.1 Generation of PWM signal .....	23
Setup Simulink with Arduino .....	23
Run the Simulation.....	23
4.1.2 For Sine Wave PWM Generation:.....	25
4.1.3 For Half Sine Wave Generation: .....	26

4.2 Generate GPS data parsing program in Simulink.....	27
4.2.1 Overview of GPS data format .....	27
GPGLL sentence .....	27
Breaking Down the GPGLL Sentence .....	27
4.2.2 Algorithm .....	29
Flowchart of Algorithm .....	30
4.2.3 Implementation of Algorithm in MATLAB .....	31
5 Conclusion .....	33
6 References .....	34

# 1 Introduction

The objective of the report is to study the operation of Simulink and MATLAB, employing these tools to replicate control system models that incorporate Arduino within a simulation environment. This allows for the modeling and simulation of systems that dynamically engage with the tangible world in real time. Utilizing Simulink to program the microcontroller contributes to a refined focus on the overarching objectives of system modeling, as the platform employs a block-based programming paradigm. Consequently, the intricate programming intricacies of each step can be avoided.

Central to this endeavor is the exploration of tools such as Simulink to develop models for real-world systems, subsequently testing their behavior prior to real-world application deployment. This approach aligns with the primary thrust of the report, encapsulating the utilization of Simulink to create models for tangible systems and assess their behavior in a controlled simulation environment, thereby optimizing their eventual real-world implementation.

## 2 Literature Review

### 2.1 Literature Survey of various Arduino SBCs

This literature survey explores Arduino Single Board Computers (SBCs), analyzing applications, customizations, and advancements. It covers official documentation, community forums, research papers, maker websites and open-source repositories, providing insights into the state-of-the-art developments up to the cutoff date. This mainly includes the microcontroller boards which are still active and not discontinued from production and service.

Arduino boards are mainly classified into 3 families: Nano, MKR and Classics.

The Nano Family comprises compact boards loaded with features, spanning from the affordable Nano Every to the advanced Nano 33 BLE Sense / Nano RP2040 Connect with Bluetooth / Wi-Fi modules. These boards include built-in sensors like temperature/humidity, pressure, gesture, and microphone. They support Micro-Python programming and facilitate Machine Learning capabilities. The Table-1 (Arduino, 2022) below displays a data excerpt pertaining to the Nano Boards that are currently active:

Family			Nano					
Board			Nano	Nano 33 BLE	Nano 33 BLE SENSE	33 IoT	Nano Every	Nano RP2040 Connect
MicroController			ATmega328	nRF52840	nRF52840	SAMD21 Cortex-M0+32Bit low power ARM MCU	ATmega4809	Raspberry Pi RP2040
Power	Operating Voltage	V	5	3.3	3.3	3.3	5	3.3
	Input Voltage (recommended)	V	7-12	5-18	5-18	5-18	7-18	5-21
	Input Voltage (limit)	V	6-20	4.5-21	4.5-21	4.5-21	7-21	6-21
	DC Current per I/O pin	mA	40	-	-	-	20	-
	DC Current for 3.3V pin	mA	50	15	15	7	50	40 (sink-28)
I/O	Digital I/O Pins		22	14	14	14	14	20
	PWM Digital I/O Pins		6	all digital pins	all digital pins	11	5	18
	Analog Input Pins (ADC)		8	8	8	8	8	8
	Analog Output Pins (DAC)		1	-	-	1	-	-
Memory	Flash Memory	kB	32 (2)	1MB	1MB	256	48	16MB
	SRAM	kB	2	0.25	0.25	32	6	520
	EEPROM		1	-	-	-	0.25	448 (ROM)
Clock Speed		MHz	16	64	64	48	20	133
LED_BUILTIN			13	13	13	13	13	13
Dimensions	Length	mm	45	45	45	45	45	45
	Width	mm	18	18	18	18	18	18
	Weight	g	7	5	5	5	5	6
Communication	UART		1	1	1	1	1	1
	I2C		1	1	1	1	1	1
	SPI		1	1	1	1	1	1
	Bluetooth		-	1	1	1	1	1
	WiFi		-	-	-	1	-	1
Sensors	IMU		-	-	1(LSM9DS1)	1 (LSM6DS3)	-	1(LSM6DSOXTR)
	Microphone		-	-	1(MP34DT05)	-	-	1(MP3DT06JTR)
	Pressure		-	-	1(LPS22HB)	-	-	-
	Gesture, Light, Proximity		-	-	1(APDS9960)	-	-	-

Figure 1: Table of Boards from Nano Family

Note. Data from Arduino Hardware, Arduino, 2022 (<https://www.arduino.cc/en/hardware>)

The MKR Family consists of boards, shields, and carriers that can be combined for projects without extra circuitry. Boards have radio modules (excluding MKR Zero) for Wi-Fi, Bluetooth, LoRa, Sigfox, NB-IoT communication. All boards use Cortex-M0 32-bit SAMD21 processor with a

crypto chip for secure communication. Shields and carriers expand functions like sensors, GPS, Ethernet, motor control, and RGB matrix. The Table-2 (Arduino, 2022) below displays a data excerpt pertaining to the MKR Boards that are currently active.

Family		MKR							
Board		MKR GSM 1400	MKR1000 Wi-Fi	MKR NB 1500	MKR Vidor 4000	MKR WAN 1300	MKR WAN 1310	MKR Wi-Fi 1010	MKR ZERO
MicroController		SAMD21 Cortex-M0+ 32bit low power ARM MCU							
Power	Operating Voltage	V	3.3	3.3	3.3	3.3	3.3	3.3	3.3
	Input Voltage (rec'd)	V	5-7	5-7	5-7	5-7	5-7	5-7	5-7
	DC Current per I/O pin	mA	7	7	7	7	7	7	7
	Battery Specs		3.7V 2500mAh	3.7V, 700mAh	3.7V, 1500mAh	3.7V, 1024mAh	2*AA Or AAA	1024 mAh	3.7V, 1024mAh
I/O	Digital I/O Pins		8	8	8	8	8	8	22
	PWM Digital I/O Pins		13	12	13	13	12	13	12
	Analog I/P Pins (ADC)		7	7	7	7	7	7	7
	Analog O/P Pins (DAC)		1	1	1	1	1	1	1
Memory	Flash Memory	kB	256	256	256	256	256	256	256
	SRAM	kB	32	32	32	32	32	32	32
Clock Speed		MHz	48	48	48	48	48	48	48
LED_BUILTIN			6	6	6	6	6	6	32
Dimensions	Length	mm	67.64	61.5	67.64	83	67.64	67.64	61.5
	Width	mm	25	25	25	25	25	25	25
	Weight	g	32	32	32	43.5	32	32	32
Connectivity	SIM Card		Micro SIM	-	Micro SIM	-	-	-	-
	Antenna Gain	dB	2	-	2	-	2	2	-
	Radio Module		u-blox SARA-U201	ATWINC1500	u-blox SARA-R410M-02B	u-blox NINA-W102	CMWX1ZZABZ	CMWX1ZZABZ	u-blox NINA-W102

Figure 2: Table of Boards from MKR Family

Note. Data from Arduino Hardware, Arduino, 2022 (<https://www.arduino.cc/en/hardware>)

The Classic Family includes boards such as the legendary Arduino UNO and other classics such as Leonardo & Micro. The Table-3 (Arduino, 2022) below displays a data excerpt pertaining to the Classic Boards that are currently active.

Family		Classic									
Board		UNO Rev3	Mega 2560 Rev3	Leonardo	Mini	Due	Micro	Zero	UNO Wi-Fi Rev3	UNO R4 Minima	UNO R4 WiFi
MicroController		ATmega328P	ATmega2560	ATmega32u4	ATmega328P	AT91SAM3X8E	ATmega32u4	ATSAMD21G18	ATmega4809	Renesas RA4M1	Renesas RA4M1
Power	Operating Voltage	V	5	5	5	3.3	5	3.3	5	5	5 (ESP32-S3 ~ 3.3)
	I/P Voltage (rec'd)	V	7-12	7-12	7-12	7-12	7-12	7-12	7-12	7-12	7-12
	I/P Voltage (limit)	V	6-20	6-20	6-20	6-20	6-16	6-20	6-20	6-24	6-24
	DC Current per I/O pin	mA	20	20	40	20	130	20	7	20	8
I/O	DC Current for 3.3V pin	mA	50	50	50	50	800 (same for 5V pins)	50	800	50	-
	Digital I/O Pins		14	54	20	14	54	20	14	14	14
	PWM Digital I/O Pins		6	15	7	6	12	7	10	5	6
	Analog I/P Pins (ADC)		6	16	12	6	12	12	6	6	6
Memory	Analog O/P Pins (DAC)		-	-	-	-	2	-	1	-	1
	Flash Memory	kB	32 (2)	256 (8)	32 (4)	32	512	32 (4)	256	48	256
	SRAM	kB	2	8	2.5	2	96	2.5	32	6	32
	EEPROM		1	4	1	1	-	1	-	0.25	8
Clock Speed		MHz	16	16	16	16	84	16	48	16	48
LED_BUILTIN			13	13	-	-	-	13	25	25	25
Dimensions	Length	mm	68.6	101.52	68.6	34.2	101.52	48	68.6	68.85	68.85
	Width	mm	53.4	53.3	53.3	26.7	53.3	18	53	53.4	53.34
	Weight	g	25	37	20	34.2	36	13	12	25	30
Communication	UART		1	4	1	1	4	1	2	1	1
	I2C		1	1	1	1	1	1	1	1	1
	SPI		-	1	1	-	1	1	1	1	1
	CAN		-	-	-	-	-	-	-	1	1
	Bluetooth		-	-	-	-	1	-	1	-	-
	WiFi		-	-	-	-	1	-	1	-	1

Figure 3: Table of Boards from Classic Family

Note. Data from Arduino Hardware, Arduino, 2022 (<https://www.arduino.cc/en/hardware>)

In addition to the boards listed in the preceding tables, there are numerous additional boards that are also compatible with Arduino. In addition, several single-board computers (SBCs) also

possess various versions. For instance, the widely used Uno R3 has undergone an upgrade to become the Uno R4. Moreover, various official and unofficial Arduino shields are also available in the market which enhance the features of existing Arduino Development boards.

## 3 Simulink

Simulink is a block diagram environment for multi-domain simulation and Model-Based Design. It supports system-level design, simulation, automatic code generation, and continuous test and verification of embedded systems. Simulink provides a graphical editor, customizable block libraries, and solvers for modeling and simulating dynamic systems. It is integrated with MATLAB, enabling us to incorporate MATLAB algorithms into models and export simulation results to MATLAB for further analysis.

### 3.1 Simulink Support Package for Arduino Hardware

The Simulink Support Package tailored for Arduino Hardware empowers the development and execution of Simulink models directly on Arduino boards. Within this package resides an array of Simulink blocks, facilitating the configuration and access of Arduino sensors, actuators, and communication interfaces. This comprehensive offering further facilitates real-time observation and refinement of algorithms formulated in Simulink, as they undergo live execution on the Arduino platform.

#### 3.1.1 Setup and Configuration

##### 3.1.1.1 Install Support Package

1. On the MATLAB® Home tab, in the Environment section, select Add-Ons > Get Hardware Support Packages.

In the Add-On Explorer window, click the support package and then click Install.

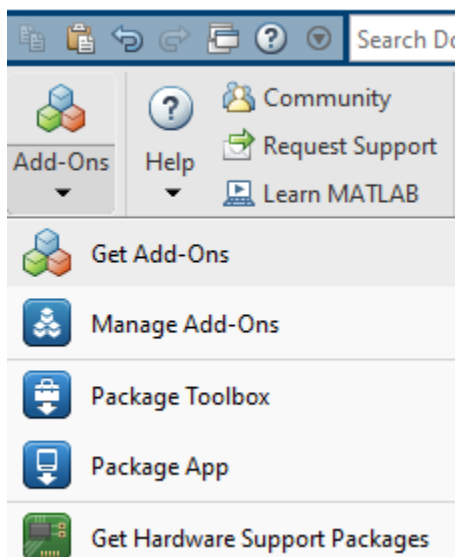


Figure 4: Install Support Package

Alternatively, the package can also be directly downloaded from the MATLAB official website and installed thereof.

### **3.1.1.2 Hardware Setup**

Hardware boards and devices supported by MathWorks® require additional configuration and setup steps to connect to MATLAB and Simulink. Each support package provides a hardware setup process that guides you through registering, configuring, and connecting to your hardware board. If the support package is already installed, you can start the hardware setup by opening the Add-On Manager.

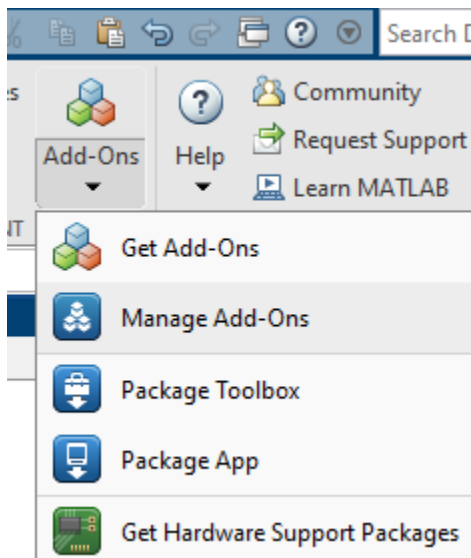


Figure 5: Hardware Set-Up

In the Add-On Manager, start the hardware setup process by clicking the Setup button and follow the instructions for configuring the support package for the hardware.

### **3.1.1.3 Open Block Library for Arduino Hardware**

Arduino Block Library can be opened from the MATLAB Command Window by typing command `>> arduinorootlib`

Or from Simulink Library Browser as



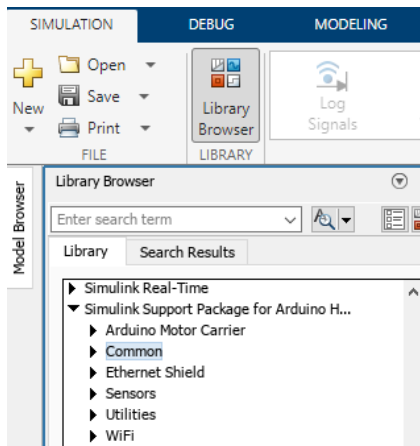


Figure 6: Simulink Hardware Support Package

### 3.1.2 Modeling

Required models can be made by Simulink code blocks and for hardware connection, add blocks from *Simulink Support Package for Arduino Hardware* to support hardware protocols.

The library contains 5 code block libraries: Arduino Motor Carrier, Common, Ethernet Shield, Sensors, Utilities, and Wi-Fi.

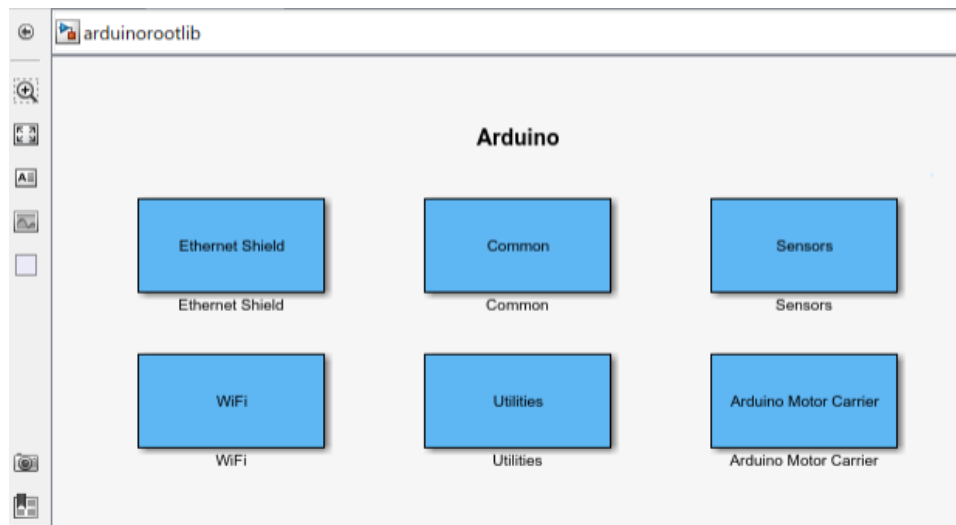


Figure 7: Block sub-libraries in Hardware Support Library

### 3.1.3 Run on Target Hardware

First, Connect the Arduino hardware to the host computer using a USB cable and create or open a Simulink model.

Then, one can prepare, configure, and run a model on Arduino hardware.

To prepare and run the model:

- 1 Use File > Save As to create a working copy of your model.
- 2 In the Simulink model, on the Modeling tab, click Model Settings to open the Configuration Parameters dialog box.
- 3 When the Hardware Implementation pane opens, set the Hardware board parameter to the specific Arduino board that is being used.
- 4 On the Hardware tab of the Simulink model, in the Mode section, select Run on board and then click Build, Deploy & Start. This action automatically downloads and runs the Simulink model on the board.

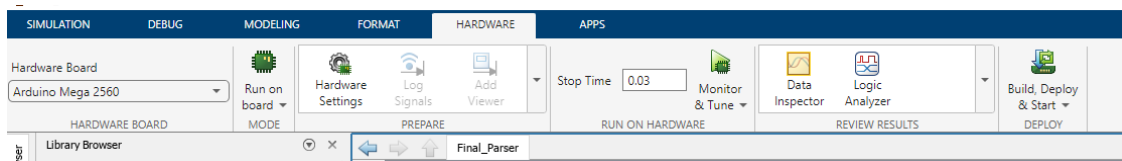


Figure 8: Run Model on Target Hardware

The Model can also be configured to be run in external mode with desired communication method, i.e., over Ethernet, Wi-Fi, or Serial.

### **Stop or Restart a Model on the Arduino Hardware**

The Arduino hardware runs the application created from the model in flash memory. The application resides in the flash memory, even after the power is disconnected from the hardware.

To stop an application running on Arduino hardware, one can:

- Disconnect the power from the hardware. When you reconnect the power, the hardware starts running the application again.
- Alternately, run a new or updated application on the hardware. This action automatically stops and erases the previous application running on the Arduino hardware.

To restart the model running on the Arduino hardware, press the RESET button on the board

### **3.1.4 Device Driver Blocks**

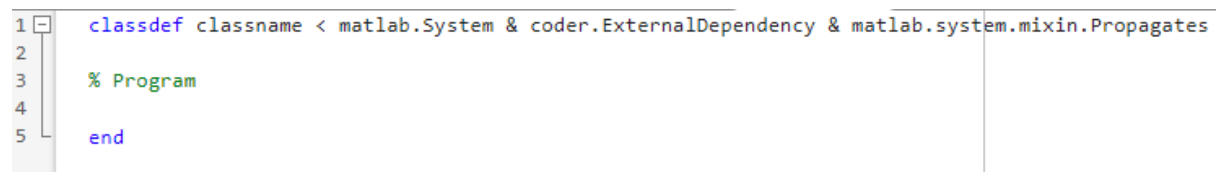
A device driver block is a specialized MATLAB System block that generates custom C/C++ device driver code when deployed to an Arduino hardware board. Device driver blocks provide easy access to hardware board features, such as communication protocols or hardware libraries, not included in the default “Simulink Support Package for Arduino Hardware”.

A Simulink device driver block can be generalized to one of two groups based on their port types: Source Blocks and Sink Blocks.

To define the behavior of the device driver block, you use a System object. Then you use a MATLAB System block to reference the object and include it in a model. Each System object uses the *setupImpl*, *stepImpl*, and *releaseImpl* methods to define the code initialization, pin output behavior, and code termination for the device driver block. Through conditional statements in the *stepImpl*, the device driver block operates in simulation mode. This mode enables the entire model to run on the host computer either when testing or when you do not have access to an Arduino. System objects also provide services for adding build artifacts. Such artifacts include source files, include paths, shared libraries, and preprocessor defines, to the Simulink generated code. These artifacts automatically define the port properties of a block and generate a block mask.

### **3.1.4.1 Class Definition**

At the top of the System Object code, you define the name of your System Object and the classes it inherits from.

A screenshot of a MATLAB code editor showing a class definition. The code is as follows:

```
1 classdef classname < matlab.System & coder.ExternalDependency & matlab.system.mixin.Propagates
2
3 % Program
4
5 end
```

Figure 9: Class Definition in Matlab System Device Driver Block

All System Objects must inherit from *matlab.System*. In addition, device driver System Objects inherit from *coder.ExternalDependency* that provides APIs to add build artifacts to generated code. The *matlab.system.mixin.Propagates* class provides APIs to define the output size, data type, and complexity of a System Object. Object can inherit from other classes, e.g. *matlab.system.mixin.CustomIcon*, which lets us specify the name and icon used by a MATLAB System Object block.

### **Constructor Method**

The standard constructor method can be given as:

```

%Methods
methods
    % Constructor
    function obj = nmea_Parser(varargin)
        % Constructor
        setProperties(obj, nargin, varargin{:});
    end
end

```

---

Figure 10: Constructor Method

### **Initialization, Output and Termination Methods**

These methods define what happens at initialization, output, and termination. Use `setupImpl` to initialize the hardware peripheral. Use `stepImpl` to read from or write to the hardware peripheral.

Use `releaseImpl` to release hardware resources used. These three methods are the backbone of defining the behavior of a device driver block.

```

methods (Access=protected)
    function setupImpl(obj)
        % Implement tasks that need to be performed only once
    end
    function stepImpl(obj,u)
        % Device driver output
    end
    function releaseImpl(obj)
        % Termination code
    end
end

```

Figure 11: I/O and Termination Methods

### **Input and Output Signal Properties**

This code section defines the number of inputs or outputs of a block and the data types and sizes. For example, the `getNumInputsImpl` method in a sink block, sets the number of input ports. Similarly, the `getNumOutputsImpl` method in a source block sets the number of outputs ports.

```

methods (Access=protected)
    % Specifies the name of input port
    function num = getNumInputsImpl(~)
        num = 1;
    end
    %-----
    % Specifies the name of input port
    function name = getInputNamesImpl(~)
        name = 'Name';
    end
    %-----
    % Specifies the number of output ports
    function num = getNumOutputsImpl(obj)
        num = 1;
    end
    %-----
    % Specifies the name of input port
    function name = getOutputNamesImpl(~)
        name = 'Name';
    end
    %-----
    % Specifies the data type of output ports
    function out = getOutputDataTypeImpl(obj)
        out = {'double'};
    end
    %-----
end

```

Figure 12: Setting Up I/O Signal Properties

## **Build Artifacts**

The build artifacts define the source file locations, include paths, shared libraries, library search paths, and preprocessor definitions required to compile the device driver code. Use the `getDescriptiveName` method to define an identification string to the System object. The code generation engine uses this string to report errors. Use the `isSupportedContext` method to specify the code generation context. In device driver blocks, only the real-time workshop (rtw) code generation context applies, so this function always specifies 'rtw'. Use the `updateBuildInfo` method to specify source and header files, include paths, libraries, and defines required to build the System object.

```

methods (Static)
    function name = getDescriptiveName()
        name = 'descriptive name';
    end
    function tf = isSupportedContext(context)
        tf = context.isCodeGenTarget('rtw');
    end
    function updateBuildInfo(buildInfo, context)
        if context.isCodeGenTarget('rtw')
            % Update buildinfo
            srcDir = fullfile(fileparts(mfilename('fullpath')), 'src'); %#ok<NASU
            includeDir = fullfile(fileparts(mfilename('fullpath')), 'include');
            addIncludePaths(buildInfo, includeDir);
            % Use the following API's to add include files, sources and
            % linker flags
            %addIncludeFiles(buildInfo, 'source.h', includeDir);
            %addSourceFiles(buildInfo, 'source.c', srcDir);
            %addLinkFlags(buildInfo, {'-lSource'});
            %addLinkObjects(buildInfo, 'sourcelib.a', srcDir);
            %addCompileFlags(buildInfo, {'-D_DEBUG=1'});
            %addDefines(buildInfo, 'MY_DEFINE_1')
        end
    end
end

```

Figure 13: Building Artifacts

This way Simulink driver code blocks which are not present in the library can be generated.

### **Write the Hardware-specific C/C++ Code**

In most cases, to integrate device driver code into a Simulink block, one needs to write a wrapper function around the API provided by the hardware vendor. Follow these steps to develop the C/C++ code required to implement the required functionality.

### **Description for Users**

Description of the block can also be added for the user by replacing comments in the beginning of the class file.

```

classdef classname < matlab.System & coder.ExternalDependency & matlab.system.mixin.Propagates
    % Class Description Here
    ...
end

```

Figure 14: Adding Description for Users

## **4 Simulation**

### **4.1 Generation of various signals using Simulink and Arduino Hardware:**

#### **4.1.1 Generation of PWM signal**

##### **Setup Simulink with Arduino**

- 1) Establish serial connection between MATLAB and Arduino board.
- 2) Create a new Simulink model and open it.
- 3) Add the required blocks to the model:
- 4) Add a "sequence Generator" block from the Simulink Library Browser. This block generates a square wave, simulating the PWM signal. And connect it to the PWM block of Simulink library.
- 5) Add the "PWM Output" block from "Simulink Support Package for Arduino Hardware" to control the digital output pin (in this case, pin D13) of the Arduino.
- 6) Connect the blocks as follows:
  - a) Connect the "Sequence Generator" block to set the parameters, such as amplitude, frequency and duty cycle.
  - b) Configure the board with Simulink under the tab Modelling -> Model Settings -> Hardware Implementation.
  - c) Set the board to appropriate board (here Nano 3.0) and port.
- 7) Save your Simulink model.

##### **Run the Simulation**

- 1) Select the "Run on Board" mode in the hardware tab to simulate the model on Arduino.
- 2) Then, select "Monitor and Tune" to initiate simulation.
- 3) The PWM signal should be generated and sent to the Arduino, which will control the PWM output accordingly.
- 4) We would observe in-built LED of Arduino turning on/off based on the duty cycle specified in the "Sequence Generator" block.

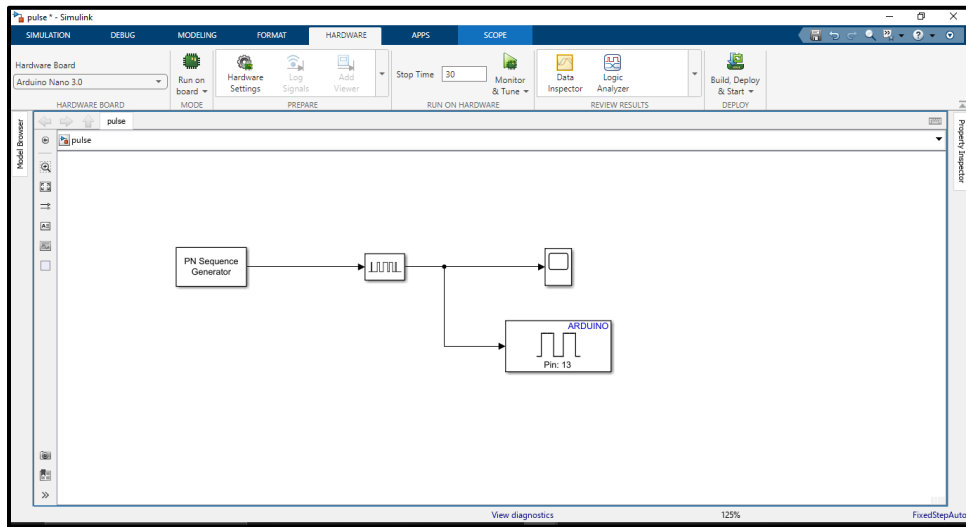


Figure 15: Model for PWM signal Generation

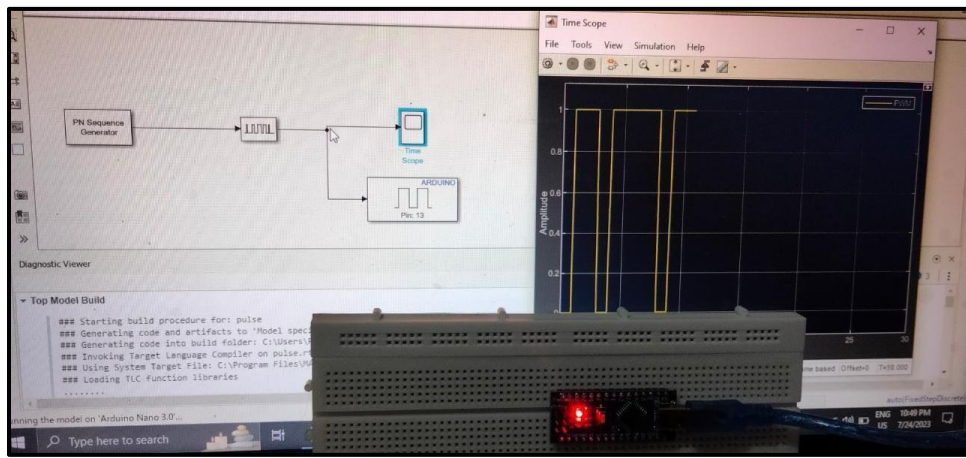


Figure 16: Running Model PWM signal Generation Model on Hardware



#### 4.1.2 For Sine Wave PWM Generation:

- 1) Select "Sine Wave" as the source from the Simulink Library Browser. And connect it to the PWM block of Simulink library.
- 2) Add the "PWM Output" block from "Simulink Support Package for Arduino Hardware" to control the digital output pin (in this case, pin D3) of the Arduino.
- 3) Then, follow the steps same as above from 5 onwards. And run the simulation.

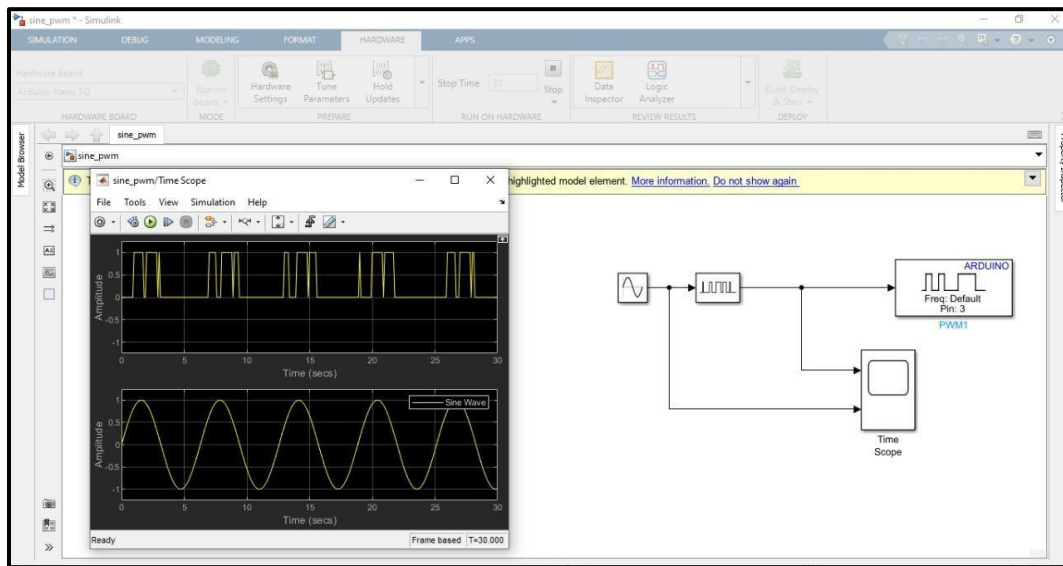


Figure 17: Model for Sine Wave PWM Generation

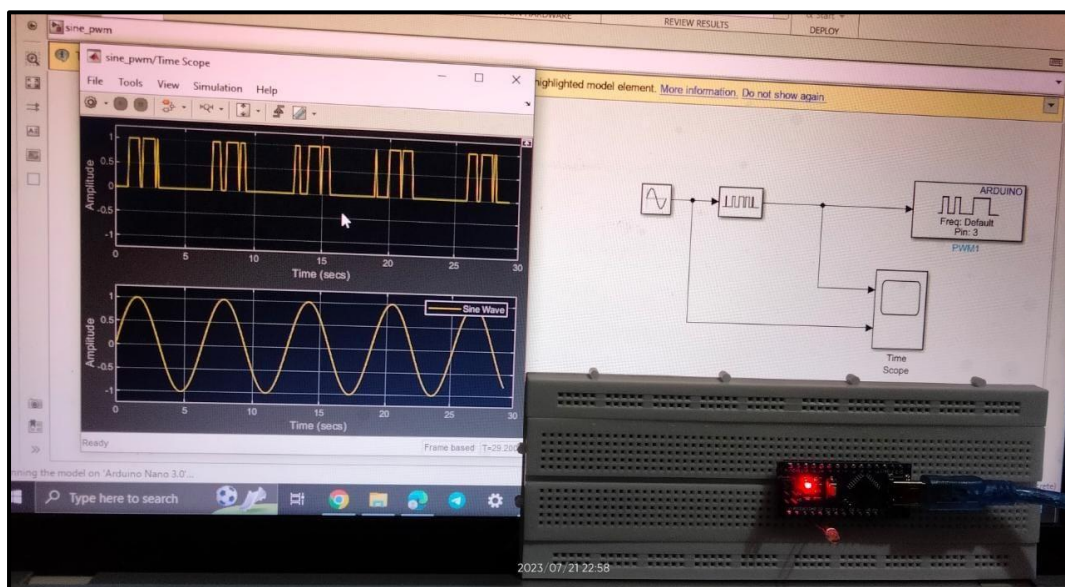
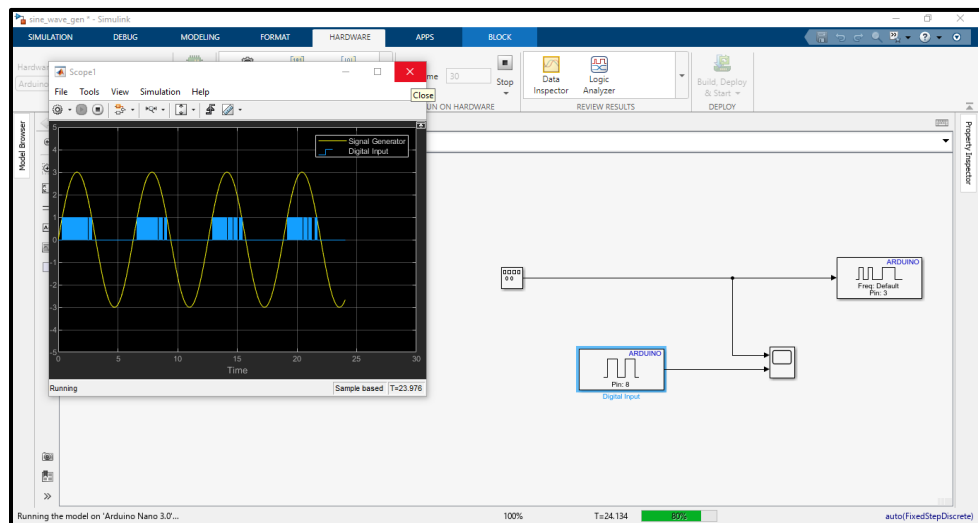


Figure 18: Running Sine PWM Model on Hardware

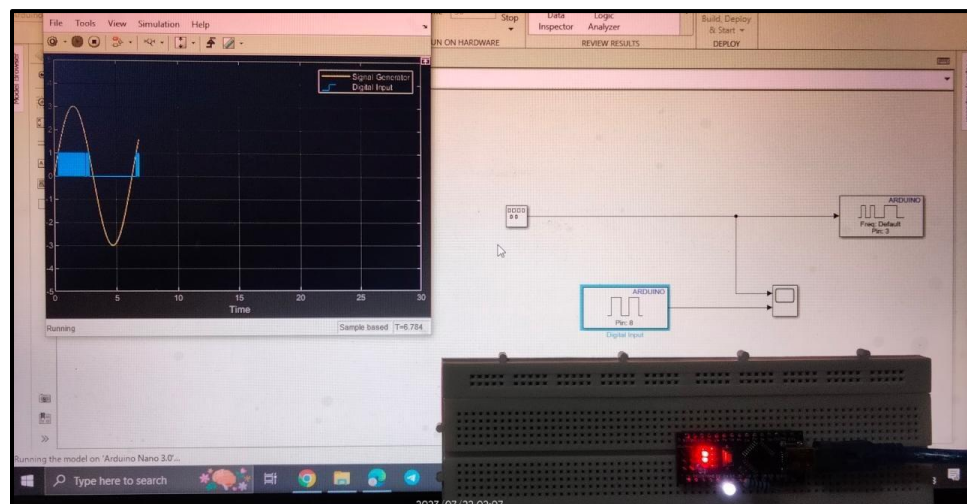
### 4.1.3 For Half Sine Wave Generation:

1. Select "Sine Wave" as the source from the Simulink Library Browser.
2. Add the "PWM Output" block from "Simulink Support Package for Arduino Hardware" to control the digital output pin (in this case, pin D3) of the Arduino.

Then, follow the steps same as above from 5 onwards. And run the simulation.



**Figure 19: Generation of Model for Half Sine Wave Signal**



**Figure 20: Running the Model on Hardware for Half Sine Wave Signal**

## 4.2 Generate GPS data parsing program in Simulink

In this context, our focus shifts to the programming of a GPS data parsing program within Simulink. The realm of Simulink offers several avenues to forge personalized code blocks, each accompanied by its inherent merits, demerits, and technical constraints. In the present context, we adopt the 'Matlab System' block within the Simulink library, harnessed alongside a collective of user-defined functions. This strategic selection affords the creation of a user interface (UI) that facilitates the incorporation of user-specified parameters. Moreover, this configuration seamlessly orchestrates the interpreted execution of Simulink's code blocks, encapsulating a multifaceted approach to programmatic functionality.

### 4.2.1 Overview of GPS data format

The NMEA format is a specification that defines how data is transmitted between various marine electronic devices. GPS receiver, a marine electronic device, also transmits data in NMEA format. The data is transmitted in a sequence called a *sentence*. Each sentence contains information, such as latitude, longitude, speed, and time, as ASCII characters. A sentence can have a maximum of 80 characters. Each sentence is independent of other sentences from the receiver.

There are various GPS data formats each with their own data and sequences. These include GPGGA, GPGLL, GPVTG, GPRMC, GPGSA, GPGSV, GPMSS, GPTRF, GPSTN, GPXTE, GPZDA, etc. Here, we are concerned mainly with the GPGGA format.

#### GPGGA sentence

The GPGGA sentence is one of the most used NMEA sentences for transmitting basic GPS fix information. It provides essential information about the current GPS fix, including the latitude, longitude, altitude, fix quality, and the number of satellites in view.

#### **Breaking Down the GPGGA Sentence**

Sentence begins with two letters to represent GPS device. For example, "GP" represents GPS device. Next 3 characters 'GGA' represents protocol type i.e., Global Positioning System fix data (time, position, fix type data). The first character is '\$' (ASCII - 36) marking the start of a sentence. Commas are used as separators for data values. Asterisk (\*) is used before checksum to indicate the checksum which is the XOR of all the data between '\$' and '\*' (excluding the two). This helps check the validity of the sentence. The end of the sentence is marked using <CR><LF> represented by '\r'.

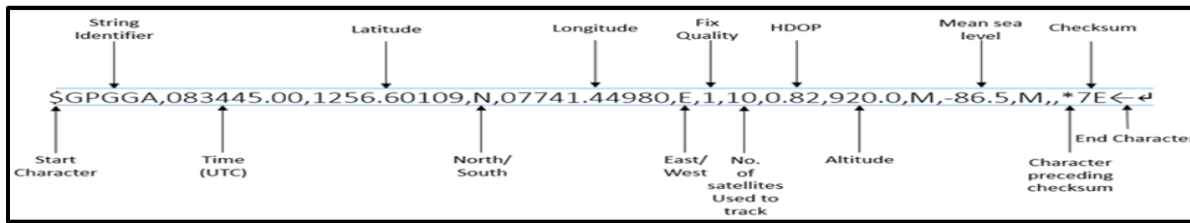


Figure 21: GPGGA String Example

Note. Data from GPS Sentences | NMEA Sentences | GPGGA GPGLL GPVTG GPRMC,RF Wireless World, 2023 (<https://www.rfwireless-world.com/Terminology/GPS-sentences-or-NMEA-sentences.html>)

Name of field	Example	Description
Message ID	\$GPGGA	GGA protocol header
UTC time	083445.00	hhmmss.sss
Latitude	1256.60109	ddmm.mmmm
N/S Indicator	N	N = North, S = South
Longitude	07741.44980	ddmm.mmmm
E/W indicator	E	E = East or W = West
Position Fix Indicator	1	'See below'
Number of Satellites Used	10	Range is 0 to 12
HDOP	0.82	Horizontal Dilution of Precision
MSL Altitude	920.0	Mean Sea Level
Units	M	Meters
Geoid Separation	-86.5	
Units	M	Meters
Age of Differential Correction		Second
Diff. ref. station ID		
Checksum	7E	XOR of values between \$ and *
<CR><LF>	\r	End of message termination

Figure 22: List of the data values given by GPGGA NMEA sentence (RF Wireless World, 2023)

Position Fix Indicator / Fix Quality

0: Fix unavailable or invalid

1: GPS SPS Mode, Fix valid

2: Differential GPS, SPS Mode, Fix valid

3-5: Not supported

6: Dead Reckoning Mode, Fix valid

#### **4.2.2 Algorithm**

- 1) Initialize a data buffer to store parsed GPS data.
- 2) Initialize a loop to continuously read incoming NMEA sentences from the GPS module.
- 3) Reset all the flags.
- 4) Read an NMEA sentence from the GPS module.
- 5) Look for startChar and when "\$" is received,
  - a) Turn storeData flag ON and start storing data in Data Buffer until checkChar "\*".
  - b) If \$ is received before \*, go back to step 3.
- 6) When checkChar is received, calculate parity (XOR of the characters received).
- 7) Store checksum from the sentence.
- 8) If checksum != parity, go back to step 3.
- 9) Parse the sentence to extract fields using a parsing function:
  - a) Split the sentence into an array of fields or separate variables based on the comma delimiter.
  - b) Extract relevant fields (e.g., time, latitude, longitude, fix quality, etc.).
- 10) Validate the extracted fields, ensuring they are in the expected format.
- 11) Perform any required data conversions.
- 12) Store the parsed data in the data buffer.
- 13) Check the buffer size:
  - a) If the buffer is full, perform real-time operations on the data (e.g., display, calculations, etc.).
  - b) Otherwise, go back to step 3 to read the next NMEA sentence.
- 14) Repeat the loop to continuously parse incoming GPS data.

### Flowchart of Algorithm

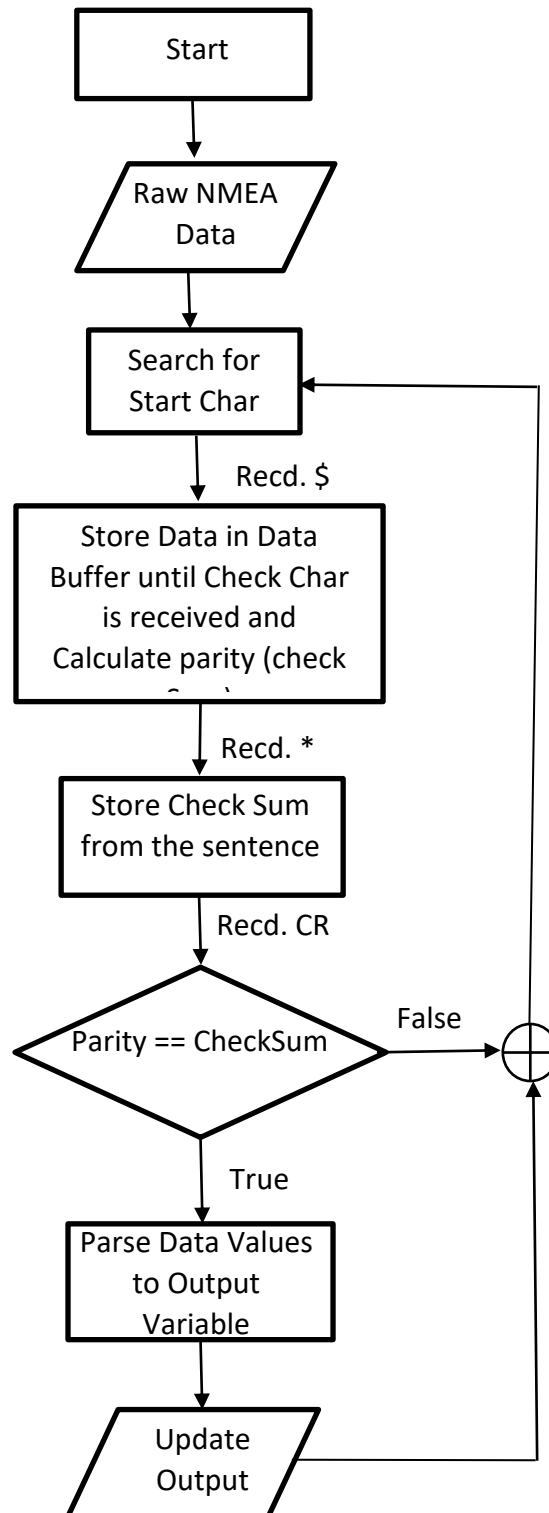


Figure 23: Flow Chart of the GPS Parser program

### 4.2.3 Implementation of Algorithm in MATLAB

The following are the Simulink 'Matlab system' blocks generated using MATLAB programming.

```

1 classdef nmea_parser < matlab.System
2     % Parses the NMEA sentence received from GPS module.
3     % This template includes the minimum set of functions required
4     % to define a System object with discrete state.
5
6     %-----Properties of NMEA Sentence-----
7     properties(Nontunable)
8
9         %Start Char - $ (dollar)
10        StartChar = uint8(36);
11        %End Char - CR (carriage return /r)
12        EndChar = uint8(13);
13        %Separator - , (comma)
14        Sep = uint8(44);
15        %Character Preceding CheckSum - * (asterisk)
16        CheckChar = uint8(42);
17        % Buffer Size
18        BufferSize = uint32(100);
19
20        %-----User entered logical values-----
21
22        % Time
23        TimeOpt (1, 1) logical = false;
24        % Latitude
25        LatOpt (1, 1) logical = false;
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Figure 24: Snapshot of Matlab Code for Driver Block

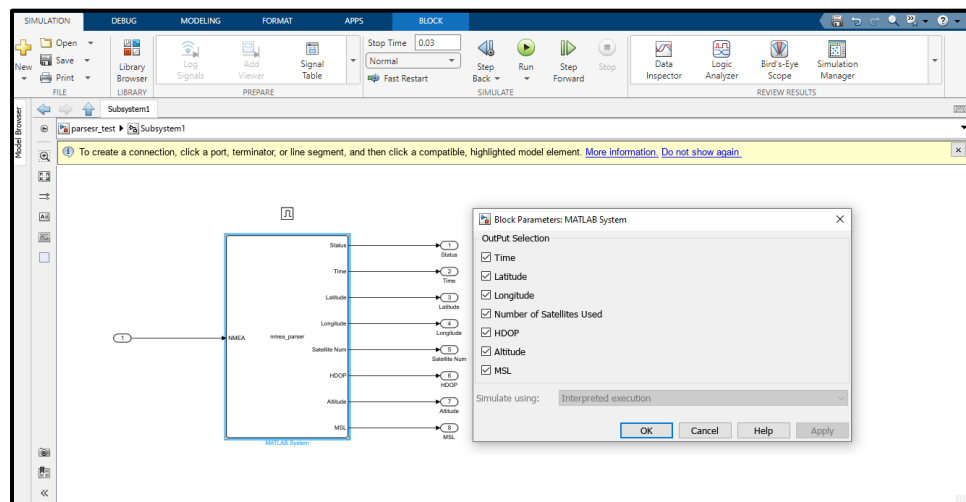


Figure 25: Simulink Block Code made using Driver Block

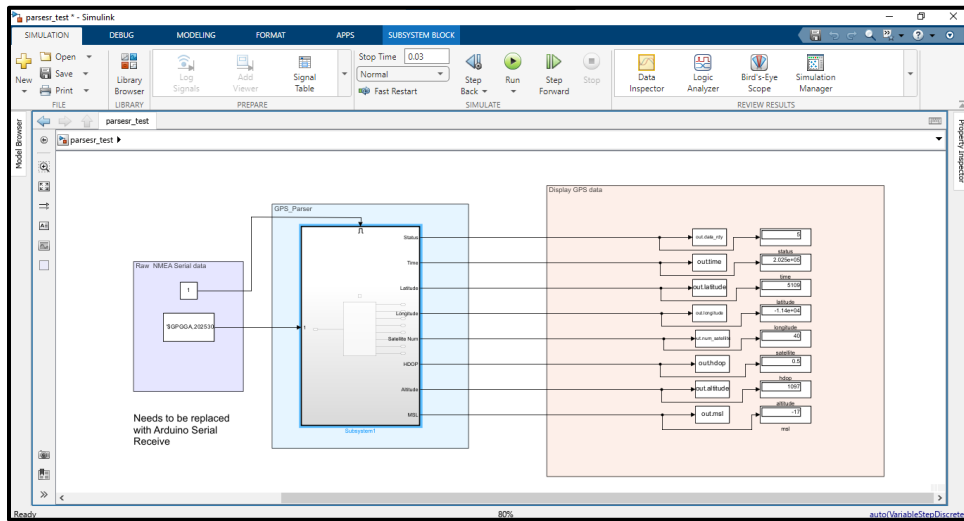


Figure 26: Testing of Code Block

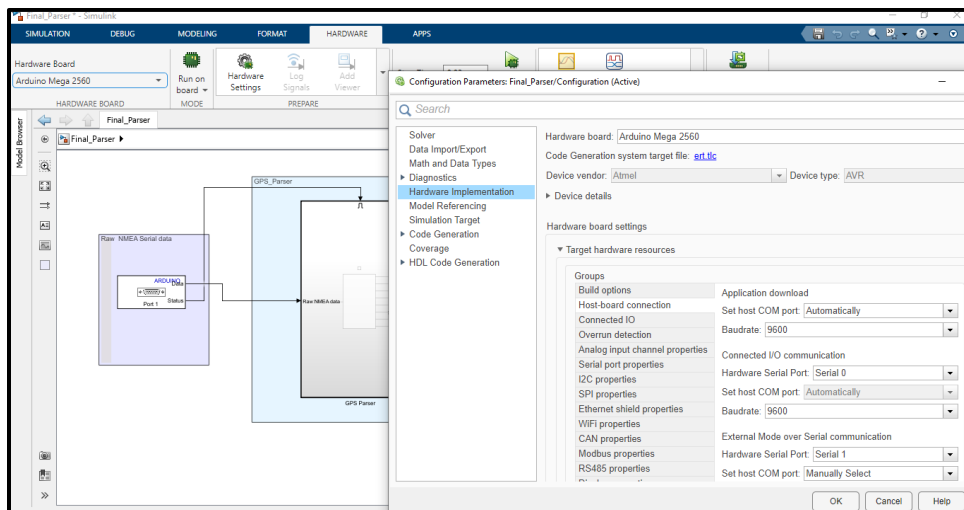


Figure 27: Hardware Configuration for the Program



## 5 Conclusion

In this report, we explored the integration of Arduino board programming with Matlab/Simulink by focusing on two key aspects: signal generation using Arduino in Simulink and developing a user defined code block, which allowed us to expand the rich library provided for working with Arduino Hardware.

We successfully demonstrated the generation of various signals using Arduino within the Simulink environment. By leveraging the Arduino support package in Simulink, we were able to design and simulate complex signal generation scenarios. This capability is invaluable for testing and verifying control algorithms, sensor interfaces, and other real-time applications. The seamless interaction between Simulink and Arduino provides engineers and researchers with a powerful platform for rapid prototyping and testing.

Another significant aspect of our exploration was the development of a GPS Parser program. We used the NMEA (National Marine Electronics Association) data format commonly output by GPS modules. Through the combination of C++ code and Matlab/Simulink blocks, we created a GPS Parser that processes NMEA sentences, extracts relevant data such as time, latitude, longitude, fix quality, and more. This parsing capability is essential for converting raw GPS data into usable information for navigation, tracking, and geo-location applications. This also helped us understand integration of C/C++ or MATLAB code to generate Simulink code blocks.

### Key Takeaways:

- The integration of Arduino with Matlab/Simulink provides a versatile platform for hardware-in-the-loop simulation, algorithm testing, and rapid prototyping.
- Simulink's Arduino support enables the creation of complex signals and waveforms for testing and validation of control systems.
- The development of a GPS Parser showcases how Matlab/Simulink can be used to process real-world sensor data, making it suitable for a wide range of applications.
- By harnessing the power of both Simulink and Arduino, engineers and researchers can streamline their development processes, reducing the time and effort required to test and implement algorithms.

In conclusion, our exploration highlights the synergy between Arduino and Matlab/Simulink, enabling efficient signal generation and the development of specialized programs like the GPS Parser. This integration empowers engineers and researchers to innovate and create solutions for diverse real-world challenges.

## 6 References

1. Arduino. (2022, April 11). *Arduino Hardware*. Retrieved July 4, 2023, from <https://www.arduino.cc/en/hardware>
2. RF Wireless World. (2023). *GPS Sentences | NMEA Sentences | GPGGA GPGLL GPVTG GPRMC*. Retrieved August 2, 2023, from RF Wireless World: <https://www.rfwireless-world.com/Terminology/GPS-sentences-or-NMEA-sentences.html>
3. The MathWorks, Inc. (2023, March). *Simulink Support Package for Arduino Hardware User's Guide*. Retrieved July 6, 2023, from Mathworks.com: <https://www.mathworks.com/help/supportpkg/arduino/>