|  |
| --- |
| Straight outta compe |
| Phase 1 Detail Design |
| Functional Breakdown of Hardware and Software Components |
|  |
| **Authors:**  **Zack Rauen**  **Juan Merchan**  **Jayesh Bokhiria**  **Eric Assenmacher** |
|  |

|  |
| --- |
|  |

# Revision History

|  |  |  |
| --- | --- | --- |
| Date | Rev. Author | Revision |
| Oct. 8, 2015 | J. Merchan | Initial layout |
| Oct. 9, 2015 | J. Merchan | Added the contents for section 3.2 |
| Oct. 9, 2015 | Z. Ruan | Added the contents for section 3.1 |
| Oct. 10, 2015 | J. Bokhiria | Added the contents for section 3.3 |
| Oct. 10, 2015 | J. Bokhiria | Revised initial layout |

Contents

[Revision History II](#_Toc432285916)

[1. Introduction 5](#_Toc432285917)

[1.1 Identification 5](#_Toc432285918)

[1.2 Purpose 5](#_Toc432285919)

[1.3 Scope 5](#_Toc432285920)

[1.4 Definitions, Acronyms and Abbreviations 5](#_Toc432285921)

[1.4.1 Definitions 5](#_Toc432285922)

[1.4.2 Acronyms 5](#_Toc432285923)

[1.4.3 Abbreviations 5](#_Toc432285924)

[1.5 References 6](#_Toc432285925)

[1.5.1 Documents 6](#_Toc432285926)

[1.5.2 Standards 6](#_Toc432285927)

[1.6 Overview 6](#_Toc432285928)

[**2.** **System Operation** 6](#_Toc432285929)

[2.1 System Description 7](#_Toc432285930)

[2.1.1 Overall Car System Schematic 7](#_Toc432285931)

[2.1.2 Block Function Categories 7](#_Toc432285932)

[2.1.3 Data Types and Port Numbers 7](#_Toc432285933)

[3. Description of Functional Blocks 7](#_Toc432285934)

[3.1 User Interface 8](#_Toc432285935)

[3.1.1 Hardware 8](#_Toc432285936)

[3.1.2 Software 9](#_Toc432285937)

[3.2 Drive Train Components 10](#_Toc432285938)

[3.2.1 Hardware 10](#_Toc432285939)

[3.2.2 Software 12](#_Toc432285940)

[3.3 Camera Data Collection and Analysis 13](#_Toc432285941)

[3.3.1 CMUcam5 Pixy Vision Sensor Camera 13](#_Toc432285942)

[3.3.2 Line Scan Camera with TAOS Linear Sensor Array (1x128 pixel image) 15](#_Toc432285943)

[3.4 Communication Encoders and Decoders 17](#_Toc432285944)

[**4. Testing of Hardware and Software Functional Blocks** 19](#_Toc432285945)

[4.1 User Interface Testing 19](#_Toc432285946)

[4.1.1 Hardware Testing 19](#_Toc432285947)

[4.1.2 Software Testing 19](#_Toc432285948)

[4.2 Drive Train Components Testing 19](#_Toc432285949)

[4.2.1 Hardware Testing 19](#_Toc432285950)

[4.2.2 Software Testing 19](#_Toc432285951)

[3.3.2 Line Scan Camera with TAOS Linear Sensor Array (1x128 pixel image) 20](#_Toc432285952)

[4.4. Communication Encoders and Decoders Testing 20](#_Toc432285953)

1. Introduction

## 1.1 Identification

This document shall be known as the System Design I and has an identifier of SOC-SysDes-1 to be referenced in future documents.

## 1.2 Purpose

The purpose of this document is to illustrate the design to the team in order to allow an efficient and quality driven implementation as well as having a sound design.

## 1.3 Scope

This documents scope is limited to the system design of the Intelligent Car Version 3. This means that implementation choices are avoided aside from the constraints already defined in *SOC-SysReq* including the use of a microcontroller. Implementations not defined there are thus avoided in this document.

## Definitions, Acronyms and Abbreviations

### 1.4.1 Definitions

* + - 1. **Car:** The main product, the Intelligent Car Version 3.
      2. **Horn:** The buzzer used on the car.
      3. **Marker:** A cylindrical pipe that represents commands to the car.
      4. **Mode:** A constant state of the system, in this case of the car. This can be changed by input from the user (1.4.1.10).
      5. **Posts:** See Marker (1.4.1.3).
      6. **Product:** The entire system (1.4.1.9) as a finished design.
      7. **System:** This refers to the entire system in use, in this case the car as well as the personal computer that the car connects to in order to use data logging. Additionally this will include the optional mobile phone that acts as a remote as well.
      8. **User:** The person(s) that will be using or interacting with the product.
      9. **We:** The members of Straight Outta CompE.
    1. Acronyms
       1. **I/O:** Input/Output
       2. **COTS:** Commercial Off The Shelf.
       3. **I2C:** Optionally I2C. Inter-Integrated Circuit.
       4. **IEEE:** Institute of Electrical and Electronics Engineers.
       5. **PC:** Personal Computer.
       6. **SPI:** Serial Peripheral Interface Bus.
       7. **SOC:** Straight Outta CompE.
    2. Abbreviations
       1. **A:** ampere(s).
       2. **Bps:** bits per second.
       3. **C:** Celsius.
       4. **kg:** kilogram.
       5. **KHz:** kilohertz.
       6. **MHz:** megahertz.
       7. **mm:** metric millimeters.
       8. **ms:** milliseconds.
       9. **nm:** nanometer.
       10. **oz:** ounces.
       11. **V:** volts.
       12. **sec:** seconds.
       13. **in**: imperial inches.
       14. **px:** pixels.
       15. **ft:** feet.
       16. **cm:** metric centimeters.
       17. **°:** degree(s).
       18. **Mag:** magnitude
  1. References

### 1.5.1 Documents

* + - 1. Material from EE416/464
      2. Figure backside of pixy camera

<http://www.hr.galagomarket.com/product_images/big/955_senzorji_seed-studio_pixy-cmucam5-sensor,-seed101990056.jpg>

* + - 1. Marker camera mounted on the pan tilt

http://www.sgbotic.com/images/detailed/9/01599-02-L1401621645538b0c8d15618.jpg

* + - 1. FRDM

http://www.element14.com/community/servlet/JiveServlet/showImage/38-17570-208127/5.jpg

### 1.5.2 Standards

* + - 1. I2C
      2. SPI
      3. IEEE 802.15
      4. UART
  1. Overview

The rest of this document discusses the operation of the system at an overall level in section 2. Also within section two are the block diagrams, one for the overall system and another for the software that shall be implemented on the microcontroller. Functional descriptions of each individual block can be found in section 3 and are separated by scope (system versus software). The last section, section 4, holds the specifications for the interfaces between the blocks as well as the specifications for COTS blocks. The COTS found in section 4 will correspond to those defined in *SOC-SysReq*.

# **System Operation**

This section includes all of the mechanical component design, interfacing of major hardware components, and all of the additional hardware components, if any, needed beyond those given. It will also include all software design of low level drivers for all major hardware components that will be interfaced with the microcontroller.

## 2.1 System Description

### 2.1.1 Overall Car System Schematic

|  |
| --- |
| F:\Senior LAB stuff\Top_Level_V3.2.1.png |

*Figure 2.2.1: System level block diagram.*

Figure 2.1.1 show the overall system logical block diagram for hardware and software. The blocks located inside the black rectangle represent software components to be implemented in the Freedom board. The other blocks located in the outside of the black rectangle represent Hardware components of the car system.

### 2.1.2 Block Function Categories

The car system block (Figure 2.1.1) diagram is divide in sub-block categories represented by the different color patterns. Each block category will be described in detail in section 3.

**2.1.2.1.** The green blocks represent the user interface components

**2.1.2.2.** The light blue Blocks represent the hardware and software components of the cars drive train

**2.1.2.3** The yellow blocks represent the data input and analysis from the two different cameras

**2.1.2.4** The orange blocks represent the different communication decoders and encoders

**2.1.2.5** The baby blue block represent the mode selection component

* + - 1. The dark blue blocks represent the data collection and storage using a SD card

### 2.1.3 Data Types and Port Numbers

Figure 2.1.1 also includes data types and pin assignments distinguish by different colors.

* + - 1. Black labels state the mane of the data type

**2.1.3.2** Red labels state the pin assignment

**2.1.3.3** Baby blue labels state the data type of each internal variable

1. Description of Functional Blocks

## User Interface

### 3.1.1 Hardware

This section describes all of the mechanical component design, interfacing of major hardware components of the LCD and User Interface. It also contains the software design which include low level drivers for all major hardware components the will be interfaced with the microcontroller.

#### Functional Specification

The base functionality of the LCD component can be found in the *SOC-SysDes-1* document. Additional specifications are laid out below.

* + - * 1. The LCD panel must accept user input by one or more buttons.
        2. The LCD panel shall indicate to the user that the input was received by updating the unit.
        3. The LCD panel will transmit data via the I2C protocol.
        4. The LCD panel must display information to the user.

#### 3.1.1.2 Signals Specifications

The SCL and SDA lines of the LCD panel require a pullup to a +5V source through a resistor; this is not provided onboard. SCL is rated for 1mA while SDA is rated for 3mA. VDD can range from 1.8V to 5.5V and the maximum current into VDD is 125mA. The timing and data transfer of these lines adhere to the I2C (100KHz, 400KHz, 1.7MHz) protocol. More information on this in section 3.4.

#### 3.1.1.3 Functional Achievement

Due to this being a Commercial-Off-The-Shelf product, the interactions with the device are more important than the inner workings. The aforementioned specifications are all achieved internally to the product itself. The only exception being displaying information to the user due to the fact that it has no information to display without external interaction. Since this product directly interacts with the microcontroller see section 3.2 for more information on interpreting user input and displaying messages.

#### 3.1.1.4 Schematic

The figure below outlines the inner connections for the LCD panel as well as the connection to the microcontroller.

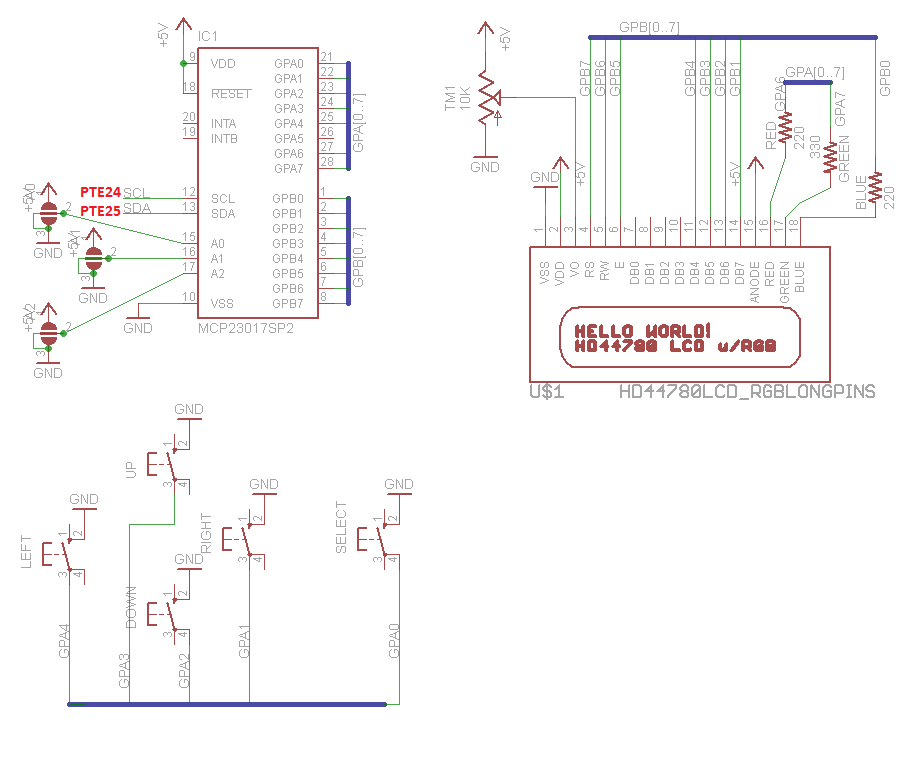


Figure 3.1.1.4: The schematic for the LCD panel.

Above, the light red text corresponds to the pins on the microcontroller. The three seemingly separate pieces connect at the blue lines to create the entire LCD panel. The top left portion manages the registers and the I2C communications. The bottom portion is the buttons for the user interface. The right portion is the LCD for displaying to the user.

### 3.1.2 Software

This describes the software design for LCD panel which is the low level drivers within the microcontroller.

#### 3.1.2.1 Functional Specification

The driver for the LCD panel has different specifications from the hardware. This is seen below.

* + - * 1. The driver must accept data from the I2C controller. (See section 3.4.)
        2. The driver must convert the raw user input to a custom data type to be used elsewhere.
        3. The driver must accept a message to be written to the LCD.
        4. The driver shall use any message given and use the I2C controller to display the message.
        5. The driver shall optimize any given message for output on an LCD display.

#### 3.1.2.2 Interface and Variables Specification

Coming in to the LCD driver from the I2C controller will be a single digit stored in a *char* type to save space. This *char* represents the button pushed by the user. Coming in from the mode controller is a *c-string*. Outgoing to the *c-string* will be a modified version of the one passed from the mode controller. Outgoing to the mode controller will be a custom data enumeration to allow representation of the different directional buttons. The type is called Button.

#### 3.1.2.3 Data Items

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Type | Variable | Value | Reason |
| 1 | int | Height | 2 | Represent height of the screen |
| 2 | int | Width | 16 | Represents 16 character width of LCD |
| 3 | typedef | Button | - | Enumerates the buttons to be used elsewhere |
| 4 | Char\* | DefaultMessage | “Currently not in a mode” | Shows the user information when there is no real information known. |
| 5 | Char\* | Break | “\n” | Predefined break that other block can use to construct their messages for pre-formatting. |
| 6 | Char\* | Buffer | “” | A buffer that allows the output to the I2C controller. |

#### 3.1.2.4 Algorithm Description

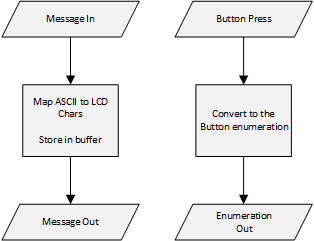
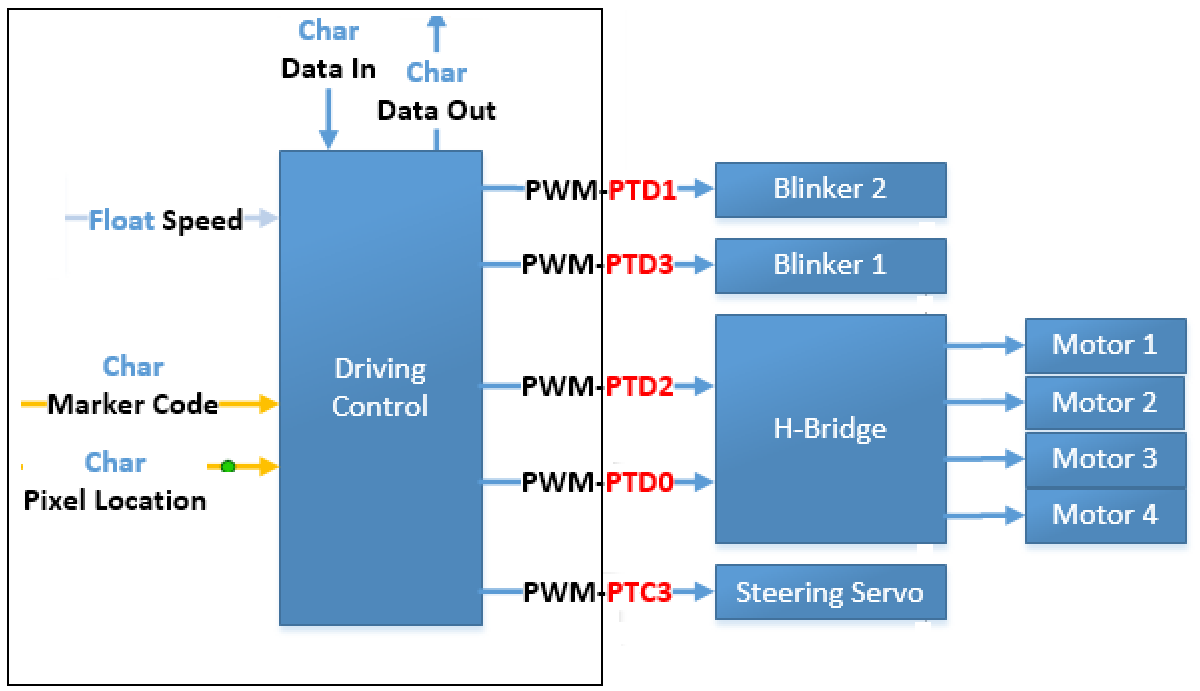


Figure 3.1.2.4: Flowchart for the algorithm within the LCD driver.

As shown in the flowchart above, the block only acts when there is a new input. Upon receiving a new

message, the message is optimized for LCD chars and sent out from the buffer. Upon getting a new button press, the corresponding Button enumeration is sent out to the mode control.

## 3.2 Drive Train Components



*Figure 3.2: Hardware and Software Components of the Car System Drive Train*

Figure 3.2 shows all the hardware components along with the one software component need it to satisfy the system requirements.

### Hardware

#### 3.2.1.1 Car Mobile Platform

The Mobile platform we will be using is a Pirate – 4WD Mobile platform from DFROBOT. The list of parts and installation procedure can be found in the ROBO3-Instruction Manual.

#### 3.2.1.2 Costume Steering design

|  |
| --- |
|  |

*Figure 3.2.1.2**Customized Steering System*

Figure 3.2.1.2 illustrates the different parts of the steering system. The parts label A, B, C, D, and E are components of the car alignment which keeps both front wheels at the same turning angle. The parts label F and G are the brackets that will hold the front motors and wheel. To make installation easier the holes for the bold (screws) are label from 1 to 8 (The each part has a matching number). The two un-label holes in parts B and C will be use to attach the Steering System to the Car Mobile Platform Described in section 3.2.1.1.

#### 3.2.1.3 Steering Servo



*Figure 3.2.1.3 Steering Servo*

An HS-422 HITEC servo will be use to steer the car by a magnitude of 30° right and 30° left for the center pivot point in both from DC motors. The Servo will use a PWN signal form pin PTC3.

#### 3.2.1.4 H-Bridge

The H-Bridge or current control of inductive load will the use to regulate the speed of all 4 DC motors. Using two PWM signals from pins PTD2 and PTD0 (Driving Controller Pins) the H-Bridge will operate two DC motors (Right side motors) with input from one PWM and the other set of motors (Left side motors) with second PWM. This will make both sides independent making it easier to execute sharper turns.

#### 3.2.1.5 DC Motors

|  |  |
| --- | --- |
| *Figure 3.2.1.5 Micro DC Geared Motor with Back Shaft* | **Electrical Characteristics:**  Operating Voltage Range: 3~7.5V  Rated Voltage: 6V  Max. No-load Current(3V): 140 mA  Max. No-load Current(6V): 170 mA  No-load Speed(3V): 90 rpm  No-load Speed(6V): 160 rpm  Max. Output Torque: 0.8 kgf.cm  Max. Stall Current: 2.8 A  Gear Ratio: 1:120 |

The DC motors will the operate using an electrical input from the H-Bridge. The H-Bridge will provide independent speeds for the left and right hand side motors.

#### 3.2.1.6 Blinkers

The blinkers are LDE light wired to the PWM pins PTD1 and PTD3 (This PWM signals are generated by the Driving Controller).

### Software

### 

#### 3.2.2.1 Driving Controller

The Driving controller uses 3 inputs to generate a series of PWM signals for to control the speed and direct of the car system.

##### **3.2.2.1.1 Speed Controller**

The speed controller uses a single-precision floating point value (float speed) to calculate the output speed ratio depending on the operation mode.

##### **3.2.2.1.2 Marker Code**

The Marker Code is an input char variable containing the marker type if there is a marker within range. Then the Driving controller will process the marker type and output PWM values depending on the magnitude of the turn or movement to be performed.

##### **3.2.2.1.3 Pixel Location**

The Pixel Location is an input char variable containing the position of the line black line with respect to the car. Then the Driving controller will process the line position and output PWM values depending on the magnitude of the turn or movement to be performed.

##### **3.2.2.1.4 Data-In**

Data-In is an input char variable containing user commands from a Bluetooth remote control. Then the Driving controller will process the user inputs and output PWM values depending on the magnitude of the turn or movement to be performed.

##### **3.2.2.1.5 Data-Out**

After processing the camera data from the Marker Code and the Pixel Location this information is transmitted using the Data-out char variable.

## 3.3 Camera Data Collection and Analysis

### 3.3.1 CMUcam5 Pixy Vision Sensor Camera

#### 3.3.1.1 Hardware

This section describes all of the mechanical component design, interfacing of major hardware components of CMUcam5 pixy vision sensor (marker) camera. It also contains the software design which include low level drivers for all major hardware components the will be interfaced with the microcontroller.

##### **3.3.1.1.1 Functional Specification**

Following are the functional specification of marker camera from Milestone 1:

**3.1.1.5** The car must use vision sensor (CMUcam5 Pixy) to detect posts

**3.1.1.6** The car shall follow the commands from the posts, located left and right side of the track (see table 3.1 of Milestone 1) at the intersection on the track

**3.1.1.7** The car must obey the post closet to the intersection when more than one popst is present

Other hardware specifications are following. The marker camera must be mounted on the pan-tilt based mechanism (Lynxmotion Pan and Tilt Kit/Aluminum). The camera must be used for all three challenges.

##### **3.3.1.1.2 Signals Specifications**

The marker camera has a power consumption of 140mA (typical). There are three ways to power the marker camera:

1. Power the marker camera through the USB cable/connector (regulated 5V)

If powering the servos through the USB cable, USB cable must be less than 4ft.

1. Power the marker camera through the I/O connector of microcontroller (regulated 5V)

Connect the pin 2 of marker camera to power source of the microcontroller and connect pins 6, 8, and 10 to ground (see figure 3.3.1.1.4.1).

Void using the ribbon cables since it has a poor current-carrying ability and will not have enough power left over to control the servos.

1. Power the marker camera through the power connector (unregulated 6V to 10V).

It is recommend by the CMUcam5 Pixy to use this option – if use pan-tilt unit with marker camera and a microcontroller without a USB cable. In this case, marker camera will be sourcing the power (source up to 1.5A of current) which will power the microcontroller and the servo.

Above method 3 for powering is might be very useful because when the microcontroller source the power to power the marker camera and pan-tilt through the power cable, either microcontroller’s power regulator will be overwhelmed, or will lose a lot of power through the cable. Nonetheless, for both either case the servo will not function probably.

Other signals are following:

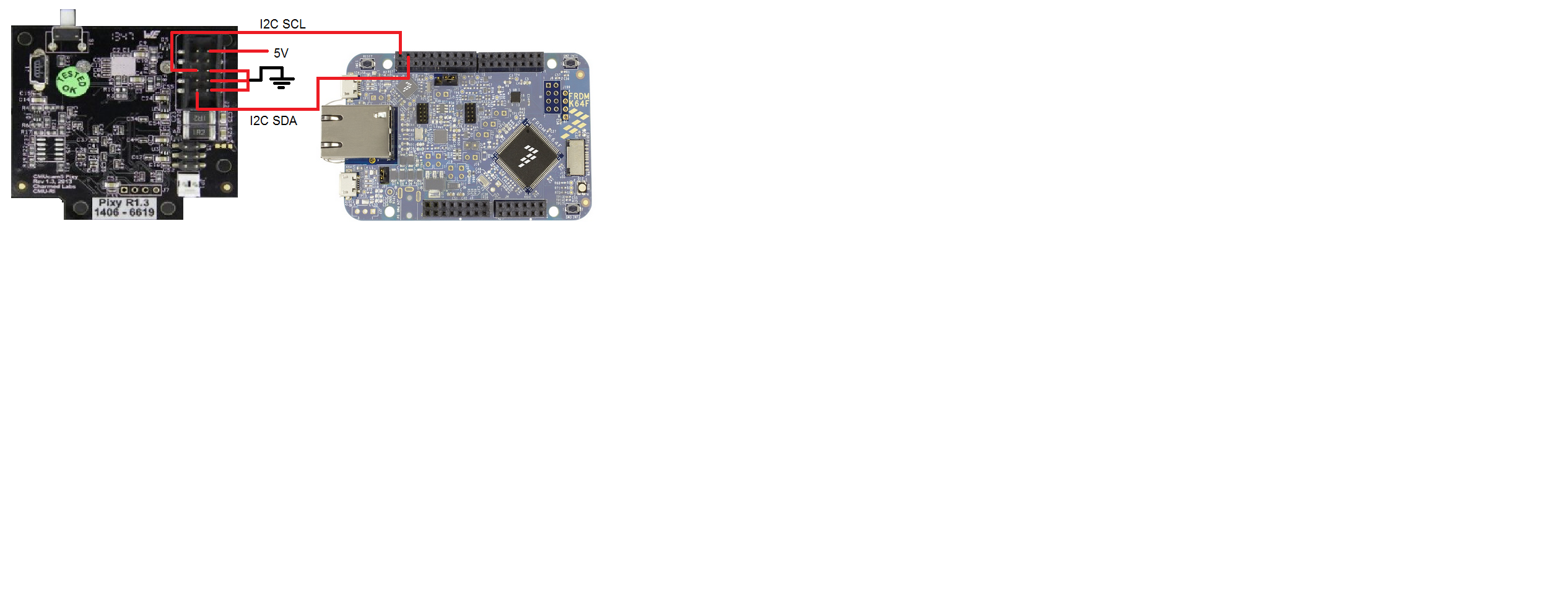
1. There are weak 4.7K pullups to 5V on SDA and SCL signals.
2. Marker camera’s digital output is 0 to 3.3V logic and can source/sink 5mA.
3. The camera’s analog (DAC) output ranges between 0 to 3.3V with roughly 200 ohm impedance.
4. The camera’s analog (DAC) output voltage is directly, linearly proportional to the object position in the image (depending on the mode)

##### **3.3.1.1.3 Function Achievement**

Mount the marker camera on pan-tilt will improve the cars ability to find markers along the track. When the camera identifies the post, the camera will process this data and detect the color pattern. Then transmit those data to the microcontroller using I2C communication standards. Based on those data the microcontroller will command the car system to determine the further action in terms of which direction the car will travel. The camera will automatically updates what it visualizes and sent that data to microcontroller every 20ms.

##### **3.3.1.1.4 Schematic**

Below figure 3.3.1.1.4.1 shows how the marker camera is connected to the microcontroller. I2C serial clock (SCL), pin 5 of marker camera is connected to the D15 of microcontroller. I2C serial data (SDA), pin 6 of marker camera is connected to the D14 of microcontroller. The pin 2, on the marker camera accepts 5V for powering the camera. This pin can be used to power the camera through the microcontroller.



*Figure 3.3.1.1.4.1: Wire connection between marker camera and microcontroller [1, 2]*



*Figure 3.3.1.1.4.4: Marker camera mounted on the pan-tilt with servo [3]*

#### 3.3.1.2 Software

This describes the software design for marker camera which include low level drivers for all major hardware components the will be interfaced with the microcontroller.

##### **3.3.1.2.1 Functional Specification**

The driver for the marker camera has different specifications from the hardware. This is seen below

Thedriver must accept data and address from the I2C controller

The driver must output marker code to the driving and data controller

Other words, the camera driver must act like a decoder which takes in the output from the I2C controller, convert it, and send it to the driving control/data controller in a format that is more useful. The Marker Code simply tells which known marker combination was detected. For more information about minor functional specifications see section 4.2.3 of Milestone 2.

##### **3.3.1.2.2 Interface and Variables Specification**

Coming in to the camera driver from the I2C controller will be a *char* type to save space. This *char* represents the data and the address from the camera. This data and address can be either line camera or marker camera. The camera driver will decode it in term of char representation of either marker code or pixel location.

The microcontroller and marker camera communicates using I2C, a two input/output signals serial data (SDA, PTE24) and serial clock (SCL, PTE25).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Type | Variable | Value | Reason |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |

### 3.3.2 Line Scan Camera with TAOS Linear Sensor Array (1x128 pixel image)

#### 3.3.2.1 Hardware

This section describes all of the mechanical component design, interfacing of major hardware components of line scan camera with TAOS sensor array. It also contains the software design which include low level drivers for all major hardware components the will be interfaced with the microcontroller.

##### **3.3.2.1.1 Functional Specification**

Following are the functional specification of line camera from Milestone 1:

**3.1.1.4** The car must use a [line] camera to detect a black line on the track.

**3.1.1.11** The car shall follow the black line on the track as closely as possible.

Following are the functional specification of line camera from Milestone 2, section 2.1.1: The car system must be able to identify its location by using the camera sensor to detect a black line on the track. Once the car system visualizes that line it must process that in data coming in and make the necessary adjustments to follow the line.

##### **3.3.2.1.2 Signals Specifications**

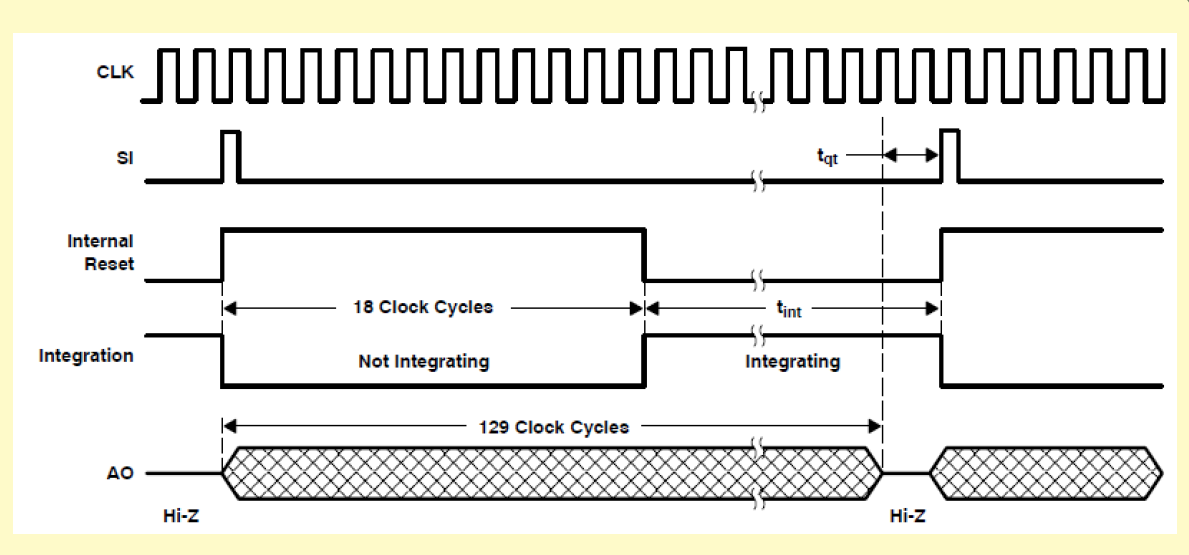
Below table represents the electrical characteristics and specification.

|  |  |
| --- | --- |
| Technical Values |  |
| Supply voltage | 3V to 5.5V |
| Max. supply current | 4.5 mA |
| High level input voltage | 2V to VDD |
| Low level input voltage | 0V to 0.8V |
| Max. high level input current | 1 |
| Max. low level input current | 1 |
| Analog output voltage (white) | 1.6V to 2.4V |
| Analog output voltage (dark) | 0V to 0.2V |
| Exposure Time | 267µS to 68mS |
| Resolution | 128 pixels |

*Table 3.3.2.1.2: Line camera’s electrical characteristics*

###### **3.3.2.1.2.1 Timing**

At the falling edge of the clock there is an SI pulse, an internal reset is high and camera is not integrating. The SI pulse goes low before rising edge of next clock pulse. During this 18 clock cycle there is no integrating which means there is no sampling. From the 19th clock cycle until the next SI pulse there is integration which means sampling. During these 129 clock cycles there is an analog output (AO) data is accumulating. After the 129th clock cycle the data is send and before the next SI pulse the data will be sent to microcontroller in order to determine the location of the line. This cycle continuously repeats during the course of car traveling on the track. As below figure 3.3.2.1.2.1shows this timing for creation and read of signals.



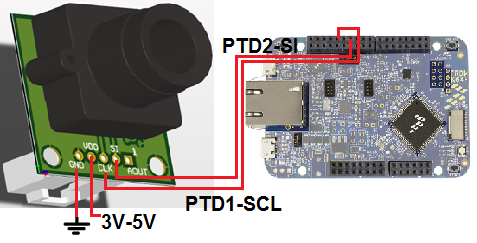
*Figure 3.3.2.1.2.1: Timing diagram of the line scan camera [Milestone 2, 4.2.4.2]*

##### **3.3.2.1.****3 Function Achievement**

Mount the marker camera on center of the car system will improve the cars ability to track the black line on the white track. The line camera will process and transmit data to the microcontroller using the SPI communication Standard. Based on those data the microcontroller will then process the data and determine were the track/ black line is in relation with the car. The camera will be mounted in the front center of car to make following the line easier.

##### **3.3.2.1.4 Schematic**

Below figure 3.3.2.1.4 shows how the line camera is connected to the microcontroller. SPI serial clock (SCL), pin 3 of line camera is connected to the D13 of microcontroller. SPI serial input (SI), pin 4 of line camera is connected to the D11 of microcontroller. The pin 2, on the line camera accepts 3V-5V for powering the camera. This pin can be used to power the camera through the microcontroller. And the pin 1 is the ground.



*Figure 3.3.2.1.4: Wire connection between line camera and microcontroller [1]*

#### 3.3.2.2 Software

This describes the software design for line camera which include low level drivers for all major hardware components the will be interfaced with the microcontroller.

##### **3.3.2.2.1 Functional Specification**

The driver for the marker camera has different specifications from the hardware. This is seen below

**3.3.2.2.1** Thedriver must accept data and address from the SPI controller

**3.3.2.2.2** The driver must output marker code and blackness to the driving and data controller.

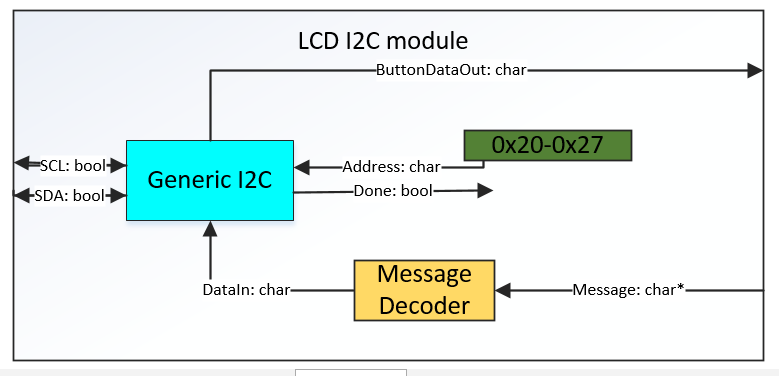
It will out output two signals Marker code and Blackness to the mode controller. In other words, the camera driver acts like a decoder which takes in the output from the SPI controller, convert it, and send it to the mode controller in a format that is more useful. More specifically, the pixel location simply tells how far the car system is off from the center line. Other words, the blackness signal (pixel location) is a representation of how black the image is, which allows the mode controller to decide how far the center of the car is from the track.

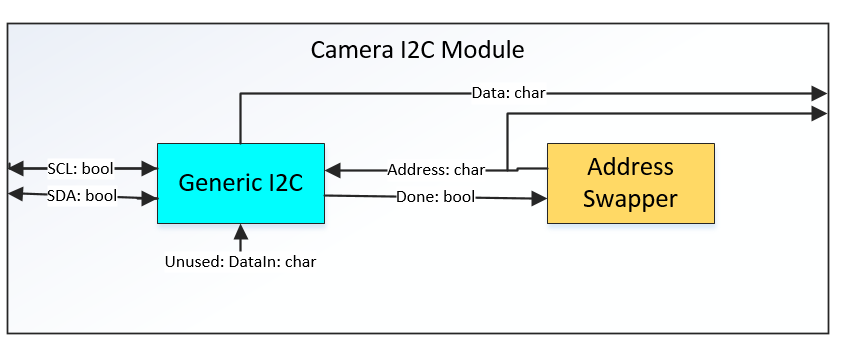
##### **3.3.2.2.2 Interface and Variables Specification**

Coming in to the camera driver from the SPI controller will be a *char* type to save space. This *char* represents the data and the address from the camera. This data and address can be either line camera or marker camera. The camera driver will decode it in term of char representation of either marker code or pixel location.

The microcontroller and line camera communicates using SPI, a two input/output signals serial input (SI, PD2) and serial clock (SCL, PTD1). A block diagram representation of this is shown in the below figure 3.3.2.2.2.1.

## 3.4 Communication Encoders and Decoders





* + 1. The I2C protocol will be used in two in two separate groups, one for camera communication, and the other for communication with the LCD and buttons
    2. The I2C communication software is split into two separate modules, with a generic I2C component designed to fit into both modules
       1. The generic I2C communication module has the ability to send and receive data using the I2C communication protocol
       2. The generic I2C takes in an address and can take in data to send out
       3. The generic I2C has a data value to store the input from sources
       4. The generic I2C has methods of letting outside sources know when the process is done
          1. The Camera I2C module has a method to swap between the addresses for each different camera
          2. The Camera I2C module passes out the Data and Address values so the camera driver can decode them
       5. The LCD I2C module hard codes the address to just the address of the LCD
       6. The LCD I2C module takes in the message from the LCD Controller and sends it out one char at a time.
       7. The LCD I2C module sends out the Button input and only reads it in when a button is pressed
    3. The UART communication is directly connected to the bluetooth adaptor
       1. The UART module sends data 8 bits at a time between a start and a stop bit
       2. The UART only sends data after the mode controller tells it to

3.4.4 Pseudocode:

Generic I2C Psudocode

Address is a char

Data is a char

done is one bit

dataOut is a char

Set Address(address)

   Address = address

SendData(Data)

   start condition

   Send Address with Write bit

   Send Data using I2C protocol

   stop condition

   set done

Receive Data()

   start condition

   Send Address with read bit

   Receive dataOut using I2C protocol

   stop condition

   set done

unset done()

   unset done

Camera I2C Psudocode

I2C is a generic I2C

CameraData is a char

Address is a char

address Select is one bit

Toggle Address()

   if address Select

   I2C.setAddress(LineAddress)

   unset address Select

   I2C.unsetDone()

   else

   I2C.setAddress(ColorAddress)

   set address Select

   I2C.unsetDone()

RunThisToGetDataNameToBeChangedLater()

   if(I2C.done is unset)

   I2C.ReceiveData()

   CameraData = I2C.data

   Address = I2C.address

   Toggle Address()

   else

   do nothing because you're impatient

LCD I2C Psudocode

I2C is a generic I2C

Button Input is a char

Message is String? need to confirm

SendMessage(Message)

   While(Message[i] is not null)

   I2C.sendData(Message[i])

   I2C.unsetDone()

Interpret Button() // run if interrupt happens

   I2C.ReceiveData()

   I2C.unsetDone()

   Button Input = I2C.data

UART Psudocode

Data Out is a Char

Rx is one bit

Tx is one bit

Pulse is 1 bit

Send() // send when pulse happens

   Tx = 0

   wait for baud of 115200

   for all of Data Out

   Tx = DataOut[i]

   wait for baud of 115200

   Tx = 1

# **4. Testing of Hardware and Software Functional Blocks**

## 4.1 User Interface Testing

### 4.1.1 Hardware Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | Procedure | Expected Results | Pass | Fail | Comments |
| 1 | Attempt to write a message to be displayed on the LCD | The corresponding message is properly displayed |  |  |  |
| 2 | Read in user input | The value corresponding to a valid button press is read in. |  |  |  |

### 4.1.2 Software Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | Procedure | Expected Results | Pass | Fail | Comments |
| 1 | Pass an integer representing a button to the block | The corresponding enumeration is output |  |  |  |
| 2 | Send a message into the block | The mapping from ASCII to LCD chars is performed correctly |  |  |  |

## 4.2 Drive Train Components Testing

### 4.2.1 Hardware Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | Procedure | Expected Results | Pass | Fail | Comments |
| 1 | Rotate the steering servo clockwise and counterclockwise | The front wheels will change direction 30° right and 30° left |  |  |  |
| 2 | Change the motor driver PWM char values (lowest to highest) | The H-Bridge changes the speeds of the left and right side motor independently |  |  |  |
| 3 | Change the Blinker PWM value | The Blinker LED’s turn on |  |  |  |

### 4.2.2 Software Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | Procedure | Expected Results | Pass | Fail | Comments |
| 1 | Switch the char Marker value between the 4 marker categories | The PWM output value variates depending on the Marker type |  |  |  |
| 2 | Change the char Pixel location for one side to an other | The PWM motor drivers should alternate to correct the location black line |  |  |  |
| 3 | User a char Marker value for Right and left turn | The servo PWM should steer in the correct direction |  |  |  |
| 4 | Change the float Speed value between the three operation modes | The PWM motor drivers should increase/ decrease the operation speed depending on the mode |  |  |  |

* 1. Camera System Components

#### 4.3.1 Hardware Testing

Below is the test procedure which validates that marker camera is working properly at the hardware level.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | Procedure | Expected Results | Pass | Fail | Comments |
| 1. | Does the marker camera recognize the color post and the car system picks the right command based on the post | At the intersection, car system’s taken action agrees to table 3.1 of Milestone 1. |  |  |  |
| 2. | Does the line camera recognize the black line on the track and follow the black line around the track | Place the car outside the track and see if the car system automatically gets on the track and follow the black line |  |  |  |

### *4.3.2. Software Testing*

Below is the test procedure which validates that marker camera is working properly at the software level.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | Procedure | Expected Results | Pass | Fail | Comments |
| 1. | Connect the marker camera to PC via USB cable and confirm that it recognize the color posts | In PixyMon camera detects the color posts |  |  |  |
| 2. | Confirm the marker camera recognize the post combinations | Place a color post in front of the car and validate with the table 3.1 of Milestone 1 |  |  |  |
| 3. | Confirm that data has been stored to the SD card | Take out the SD card from the microcontroller and plug it to the computer |  |  |  |

## 4.4. Communication Encoders and Decoders Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | Procedure | Expected Results | Pass | Fail | Comments |
| 1 | I2C controller tries to receive data from a bad address | No reaction, possibly pass info to UART to verify while NOT using I2C to verify I2C |  |  |  |
| 2 | UART sends a hello world message to a putty terminal on a computer through Bluetooth | putty says “Hello World” |  |  |  |
| 3 | In combination with the LCD test, When a button is pushed, putty is updated on what mode we are in | Putty to say Exploration in Exploration, etc. |  |  |  |