

Interface Control Document

Rampaging Chariots Grad Team 2015

Autonomous Chariot

2015

# Introduction

This document defines the interfaces between the sub-systems of the 2015 Graduate Automated Rampaging Chariot electronics.

# List Of Addenda

|  |  |
| --- | --- |
| Addendum 1 |  |
| Addendum 2 |  |
|  |  |

# SYSTEM OVERVIEW

The 2015 Graduate Automated Rampaging Chariot system is designed to enable a Rampaging Chariot to successfully navigate the RC assault course without human input.

The system achieves this by:

## Sensory Information Input

Receiving sensory information inputs from:

* Aerial view camera
* Inertial Measurement Unit (IMU)
* Ultrasonic Rangers

## Situational Awareness

Processing sensory information to provide a situational awareness map containing:

* Arena boundaries
* Location of chariot within arena
* Heading of chariot
* Obstacle location and size within arena
* Distance to objects from all sides of chariot

## Tactical Processing

Processing all the situational awareness information to make tactical decisions:

* Route to take to navigate obstacles
* Actions to take to achieve assault course objectives e.g. “kick” football
* Required motion to take to carry out the route/actions

## Control Chariot Motion

Command the master board to make the motors move the chariot to the required position and orientation with feedback from the situational awareness (e.g. turn 90 degrees, move forwards 1m)

These functions are partitioned between the sub-systems as described in the following section.

# SUB-SYSTEMS

The system comprises several sub-systems:

## Master Board



The Master Board is provided by the Rampaging Chariot Guild and provides overall low-level control of the left and right motor(s) in response to inputs of lateral and longitudinal chariot speed commands from:

1. Two channel (lateral and longitudinal speed) 2.4GHz Radio Control system providing manual control over the chariot.
2. A microcontroller sending commands to control the lateral and longitudinal speed via a serial port to the master board

The mixing of the lateral and longitudinal speed inputs is performed on the PIC on the board and the overall speed requirement for each motor is controlled via an H-bridge driver chip that modulates the speed of the motors via PWM control.

### Operation Modes

The board can be configured for manual control (via a 2.4GHz Radio Control system) or autonomous control (serial port control) by use of a configuration jumper on the board.

When the jumper is inserted, the board will be in autonomous mode and will ignore inputs from the 2.4GHz receiver.

When the jumper is removed, the board will be in manual mode and will ignore inputs from the serial port.

For this project, the system shall be configured in Autonomous mode and the serial port connected to an mbed microcontroller. See Section 3.2 for more information.

### PIC Software

The PIC shall be the Auto v1.1 PIC which has a 50000 baud rate, non-inverting serial port configured in 8N1 format for motor control commands.

### Interfaces

#### Electrical

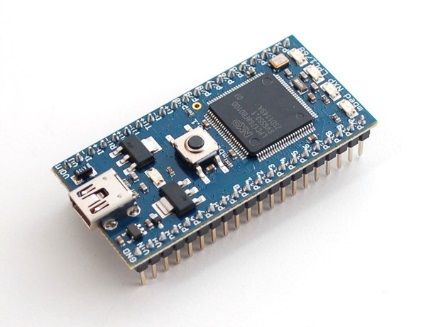
* The Master Board power is shall be supplied direct from the 18VDC on-board drive battery.
* The current draw of the board shall not exceed 20A with two motors connected in parallel to the motor terminals.

#### Communication

The Master Board has a 50000 baud rate, non-inverting serial port configured in 8N1 format that receives two bytes from the mbed microcontroller: A longitudinal and lateral power byte. The Master Board mixes these two speeds to determine the speed and direction required by each side’s motor(s).

## Mbed Microcontroller System

### Mbed LPC1768 Module



#### Overview

The mbed LPC1768 is a 96MHz ARM Cortex M3 microcontroller designed to work with C++ compiled code.

The mbed microcontroller acts as the low-level master board controller and ultrasound ranger controller. It is a slave to the Tactical Processor (TP).

Its purpose is to handle the low-level communications between the master board controller and the three GPIO14 HC-SRF04 controllers so that the Tactical Processor and Image Processor do not have to directly and so freeing valuable processing time.

It also provides manual control of the motors to move the chariot to the required starting position, emergency stop capability and debug information via a wireless Zigbee data link to the control laptop.

It supports presents a serial terminal menu system and command prompt interface to the control laptop and also features a configuration file setup.

#### Software

The mbed shall run the RC Core mbed software (for motor control) and additional software specifically written for the challenge.

The software shall be compiled and source-controlled via the online mbed compiler and source-code management suite.

#### Interfaces

##### Electrical

The mbed shall be powered by an on-board 5V power supply.

##### Communication

###### To Master Board

The mbed shall communicate unidirectionally with the Master Board via a 50000 baud rate, non-inverting serial port configured in 8N1 format.

It shall send the required lateral and longitudinal power/direction bytes via the interface, repeating them a minimum of every Xms for the duration that motion is required.

Cessation in sending the bytes or exceeding the Xms repeat limit will cause the Master Board to time out and stop the motors.

###### To/From TP (Addendum 2)

The mbed shall communicate bi-directionally with the TP via a 50000 baud rate, non-inverting serial port configured in 8N1 format.

The mbed shall request the Master Board to move the chariot in a specified direction (forwards, backwards, turn) and speed until commanded to stop by the TP.

The mbed shall request the Master Board to move the chariot forwards or backwards until a specific ranger has increased or decreased in reading by a set amount when commanded to do so by the TP.

The mbed shall request the Master Board to emergency stop and cease all communication with the master board when it receives an emergency request stop from the TP.

The mbed shall return the distance readings of all rangers when requested to do so by the TP.

Full definition of the interface messages is contained within Addendum 2.

###### To/From Control Laptop Via XBee Module



The mbed shall communicate bi-directionally with the control laptop via a 50000 baud rate, non-inverting debug serial port configured in 8N1 format connected to the wireless XBee Module in order to:

* Report debug information (errors, status updates etc.)
* Action and respond to commands from the control laptop

A wireless XBee Module connected to the control laptop end shall transmit commands and receive data to/from the mbed via a serial terminal emulator.

### HC-SRF04 Ultrasonic Rangers



#### Overview

Six of these ultrasound rangers are placed around the chariot: Forward, Forward Right, Rear Right, Rear, Rear Left and Forward Left.

The rangers are commanded to initiate measurements by their associated GPIO14 controller chips.

There is one GPIO14 controller per pair of rangers (three in total).

#### Performance

The rangers provide a maximum range 4m range and 1cm resolution. They will detect a 3cm diameter stick and 2m+.

#### Interface

##### Electrical

###### Power

The rangers require a 5VDC power supply and draw a maximum current of 30mA.

###### Digital GPIO

The rangers are operated by the Trigger (i/p) and Echo (o/p).

A high i/p on the Trigger (Trig) pin for > 10us causes the ranger to output 8 ultrasonic bursts and set its echo pin high (5VDC).

When the ranger detects the echo, it sets the echo pin low (0VDC) and thus the 5V pulse on the Echo pin has a width equal to the time the echo took to return.

The Trigger (Trig) and Echo pins are connected to the sensors GPIO14 controller chip which measures the length of the echo pulse and converts this to a distance in cm for return to the mbed when polled.

### GPIO14 HC-SRF04 Controllers



#### Overview

Three of these controller chips are installed in the chariot, one per pair of HC-SRF04 sensors.

The controllers command the rangers to perform a range measurement when commanded to do so by the mbed.

They handle the signal processing of the echo signal and when polled by the mbed, return the range measured by each of the two rangers.

#### Interfaces

##### Electrical

Each GPIO14 controller requires 5VDC and draws ~2mA.

##### Digital GPIO

Each GPIO14 controller has two Trig pins and two Echo pins, one Trig and Echo pin per ranger (A and B).

##### Communication

The controllers communicate as slaves on an I2C bus between the controllers and the mbed (Master).

Each mbed shall be configured to have the following (7-bit) unique I2C address:

|  |  |
| --- | --- |
| **Controller** | **Address (7-bit)** |
| Forward/Rear | TBD |
| Left | TBD |
| Right | TBD |

## Raspberry Pi Tactical Processor (TP)

### Raspberry Pi 2

#### Overview

##### Start-up sequence (SUS)

When the TP starts, it shall wait for the IP to send the SUS data.

Upon confirmation of successful SUS data transfer, the TP shall proceed to next stages.

#### Interfaces

##### Wi-Fi

Each Raspberry Pi shall use a compact Wi-Fi dongle (e.g. Edimax EW-7811UN), connected via an on board USB port.

#### Performance

### Inertial Measurement Unit (IMU)

## Raspberry Pi Image Processor (IP)

### Raspberry Pi 2

#### Overview

##### Start-up sequence (SUS)

When the software starts, the IP shall send run the following SUS:

* Obstacle map – A 2D Boolean array, where true signifies an obstacle and false signifies open ground.
* Chariot coordinates – using same axes as the obstacle map.
* Chariot orientation – using the same axes as the obstacle map, where north
* Football goal coordinates
* Football coordinates
* Paddle coordinates
* Ramp coordinates
* Barrel coordinates
* Barrel post coordinates

##### SUS Confirmation Request

After start up sequence is complete, the IP shall send a SUS confirmation request, proceeding with the following sequences upon receipt of a successful SUS data transfer.

##### IP Operational Loop

Once the SUS stages are complete, the IP shall run a continuous loop to send the following data:

* Chariot coordinates – using same axes as the obstacle map.
* Chariot orientation – using the same axes as the obstacle map, where north points towards top of obstacle map.
* Football coordinates
* Barrel coordinates

#### Algorithms

##### Obstacle Map

1. Get RGB image from camera
2. Convert to greyscale using a specific colour. E.g. high intensities of yellow will be set to 255, low intensities of yellow (and other colours) will be set to ~0.
3. Apply filters. E.g. median filter
4. Apply threshold to generate binary image (2D Boolean array). High intensities will be set to 1, low intensities set to 0.

#### Interfaces

##### Wi-Fi

Each Raspberry Pi shall use a compact Wi-Fi dongle (e.g. Edimax EW-7811UN), connected via an on board USB port.

#### Data Summary

|  |  |  |
| --- | --- | --- |
| **Message** | **Data Info** | **Size (Minimum)** |
| Obstacle Map | 2D Boolean array (1920x1080?) | 260KB (uncompressed) |
| Chariot coordinates | Same axes as obstacle map.  (11 Bits for each axis) | 22 Bits |
| Chariot orientation | 0° - 360° | 9 Bits |
| Football goal coordinates | Same axes as obstacle map.  (11 Bits for each axis) | 22 Bits |
| Football coordinates | Same axes as obstacle map.  (11 Bits for each axis) | 22 Bits |
| Paddle coordinates | Same axes as obstacle map.  (11 Bits for each axis) | 22 Bits |
| Ramp coordinates | Same axes as obstacle map.  (11 Bits for each axis) | 22 Bits |
| Barrel coordinates | Same axes as obstacle map.  (11 Bits for each axis) | 22 Bits |
| Barrel post coordinates | Same axes as obstacle map.  (11 Bits for each axis) | 22 Bits |
| **COORDINATES** | **22 Bits. Split into two axes.**  **E.g. Message: 1111000000010000111000**  **Split: X = 11110000000; Y = 10000111000;**  **Convert into decimal: X = 1920; Y = 1080;** | |
| **ORIENTATION** | **9 Bits. Convert binary straight into decimal.**  **E.g. 010000111 = 135 degrees** | |

#### Packet

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bytes** | **6** | **6** | **2** | **1** | **24** |
| **Field** | **MAC Destination** | **MAC Source** | **Length** | **Identifier** | **Payload** |

MAC Destination: AA-AA-AA-AA-AA-AA

MAC Source: BB-BB-BB-BB-BB-BB

Identifier: 1 for Data Transmission

2 for startup sequence confirmation request

#### Payload Dissection

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Bits** | **Word 0** | **Word 1** | **Word 2** | **Word 3** | **Word 4** | **Word 5** |
| **0** | Chariot Coordinates | Football Coordinates | Barrel Coordinates (Bits 10-21) | Barrel Post Coordinates | Football Goal Coordinates (Bits 10-21) | Paddle Coordinates (Bits 12-21) |
| **1** |
| **2** |
| **3** |
| **4** |
| **5** |
| **6** |
| **7** |
| **8** |
| **9** |
| **10** | Ramp Coordinates |
| **11** |
| **12** | Spare | Spare |
| **13** |
| **14** |
| **15** |
| **16** |
| **17** |
| **18** |
| **19** |
| **20** | Paddle Coordinates (Bits 0-11) |
| **21** |
| **22** | Chariot Orientation | Barrel Coordinates (Bits 0-9) | Football Goal Coordinates (Bits 0-9) |
| **23** |
| **24** |
| **25** |
| **26** |
| **27** |
| **28** |
| **29** |
| **30** |
| **31** | Spare |

### Raspberry Pi Camera

#### Overview

#### Interfaces

The camera shall connect to the Raspberry Pi via the on board camera port.

#### Performance

## Control Laptop

### SSH Terminals

### Serial Console

### Zigbee Module

# SYSTEM DIAGRAM