**Hardware and Software Specification for the**

Version 2 18/11/15

**Autonomous Rampaging Chariot**

1. **System Overview**

**The Raspberry Pi Computer/ Microprocessor**  controls the Autonomous system, houses the Course Model and undertakes:

Navigation Estimate

Next Move

Co-ordinate Conversion

Motor Drive

Sensor Interface

Sensor Calibration

Navigation update

**The Rampaging Chariot Master Motor Control Board**  controls one drive motor and outputs commands to the Slave Motor Control Board which drives the second motor. Control commands of Longitudinal Speed and Lateral Turn Rate are provided via a UART serial interface from the Raspberry Pi.

A switch selects between Normal Manual Radio Control and Autonomous operation. The standard input logic for catching glitches together with motor output limiting and a linear filter helps to keep the motor gear teeth attached.

In Autonomous mode the Radio Receiver provides the On/Off Command/Kill Switch

**The Sensors Interface Board** is provided to handle time critical tasks that could monopolise the computing resources of the Raspberry–Pi and delay the execution of the main navigation & control programme.

The Sensors Interface Board converts sensor commands from the Raspberry Pi, sent via an I2C serial link, into the pulse sequences required to drive a stepper motor and a servo. It also converts analogue I-R range into a digital format and converts pulse width signals from the two odometers and the ultra-sonic range sensor into a digital format.

The Sensor Interface Board contains a two-way voltage level shifter (between 3.3v and 5v), a stepper motor driver and a PIC (Programmable Interface Controller). The PIC undertakes:

The Slave I2C serial interface

Sensors sampling

Analogue to Digital conversion (A2D)

Stepper Motor 4 Phase coil logic outputs

PWM (Pulse Wave Modulation) generation

PWM to Digital conversion

If required, the Raspberry Pi can control several Sensor Interface boards via the I2C interface.

**Bluetooth**

**Telemetry**

**To PC**

**Raspberry Pi**

**Sensors Control**

**Motor Control**

**R-Pi On/Off**

**5v**

**5v**

**3.3v**

**Sensors On/Off**

**PIC =** Programmable Interface Controller

**PWM =** Pulse Wave Modulation

**UART =** Universal Asychronous

Receive/Transmit Interface

**I2C =** Inter-Integrated Circuit Interface

**UART Serial 5v**

**Level Shifter**

**PIC**

**I2C Interface**

**Sensors**

**Odometers**

**Motor Interface**

**Stepper MotorDriver**

**Stepper Motor**

**I/R**

**Range**

**Ultra Sonic**

**Range**

**Analogue**

**4x Phase**

**PWM**

**Servo Motor**

**Sensor Interface**

**5v**

**PWM**

**PWM**

**UART Serial 3v**

**I2C Serial 3v**

**System Diagram for the Autonomous Chariot - Chassis**

**Motor Control**

**(UART Serial 5v)**

**Rampaging Chariot Chassis**

**Master**

**Motor**

**Control**

**Board**

**R-Pi Power**

**Supply 5v 3A**

**Autonomous**

**/Manual Sw**

**Slave**

**Motor**

**Control**

**Board**

**PIC**

**Rx**

**18v Battery**

**18v Battery**

**Odometer**

**PIC**

**Odometer**

**Motor**

**Motor**

**Wheel**

**Wheel**

**Wheel**

**Wheel**

**Odometer**

**(PWM)**

**Power**

**(5v DC)**

**Course Model**

**Nav Estimate**

**Next Move**

**Co-ordinate Conversion**

**Motor Drive**

**Sensor Interface**

**Sensor Calibration**

**Nav Update**

**PWM**

1. **Rampaging Chariot Motor Control**

**Manual and Autonomous Modes**

In the **Manual Mode** the Master board receives longitudinal and lateral commands as two pulse width inputs from the receiver. The commands are shaped and limited as in a normal master board containing V4.0 or later software.

In the **Autonomous Mode** the Master board receives longitudinal and lateral commands as two 8 bit bytes via a serial interface. There is no shaping of these inputs, but a central dead-band is retained.

**Serial Interface**

In the Autonomous Mode the Master board receives longitudinal and lateral commands as two 8 bit bytes via a serial interface.

The PIC is a 5v microprocessor. Output logic states should be <0.8v for Low and between 2.7v and 5v for High. A bi-directional level shifter on the Sensors Interface Board is used to change between the 3.3v of the Raspberry Pi and the 5v of the Motor Drive Board.

Only the ground and data-in pins from the Master Motor Control Board are required to be connected.

The UART (Universal Asynchronous Receiver Transmitter) Serial Interface protocol is 8-N-1 which means it uses 8 data bits, No Parity and 1 Stop bit. True Logic.

The Baud Rate is 9600 bits/sec. This could be changed in software to a maximum of 50000 bits/sec

The control input is two consecutive bytes (8 bits each). Byte one is the demanded longitudinal speed and byte two is the demanded lateral turn rate (This is similar to the Channel 2 and Channel 4 outputs from the receiver).

The zero datum is 127 decimal. 127+x demands increasing positive speed. 127-x demands increasing negative speed. Maximum forward speed or right turn rate is 254 and maximum backwards speed or left turn rate is 1. Values outside these limit, e.g. zero, are invalid.

**Safety**

Fail safe logic is incorporated such that If a valid serial input is not received within 60ms an error count is generated. After 3 consecutive error counts the robot will come to a controlled stop within 1 second and after 8 invalid counts the motor drive board will shut down until a valid signal is received. It should recover to normal operation if a valid signal is received.

To help prevent inexperienced programmers generate a software overflow that could cause a full power runaway, any two consecutive control values of <44 integer and >212 integer respectively OR >212 integer and <44 integer, will cause an immediate shutdown. Recovery from this dangerous state will require the master board power to be cycled Off – On.

Programmers should therefore ensure that the R-Pi microprocessor is fully booted before switching on the Rampaging Chariot Power.

Your programme should initialise with control values of 127, 127. (Stationary)

When demanding a change in longitudinal direction, insert an intermediate demand value close to 127 bits.

e.g. Successive values of 127, 254, 254, 254,.are OK.

127, 254, 254, 254, 254, 1, is invalid.

127, 254, 254, 254, 254, 127, 2, are OK.

In Autonomous mode the standard radio control transmitter and receiver provides an on/off signal to the Master Motor Control Board. Moving the channel 2 control stick fully upwards provides an ON/GO signal. The central and down stick positions provide the OFF/STOP signal. This On/Off signal is also relayed to the R-Pi through the voltage level shifter on the Sensors Board. Using a TTL or UART connection TBD

1. **Sensor Control**

A Range Sensor measures the distance between the sensor and an obstacle in the direction it is pointing.

You can change the direction it points by

1. Turning the robot chassis left or right
2. Mounting the sensor on the shaft of a stepper motor or servo motor and rotating that shaft to the required angle
3. A combination of moving the chassis and the sensor motor shaft.

Each sensor is controlled according to the control byte sent over the I2C interface.

This bytes tells the interface board which sensor you wish to address and what it is required to do. The interface board will then provide the appropriate electrical output signals to move the sensor motor and obtain the sensor range or odometer distances.

Every type of sensor and every type of motor has different characteristics and the interface board knows how to convert your standard commands for the particular type of motor, gearing and speed of rotation. It will also convert analogue range values into digital values for transmission to the R-Pi over the I2C serial link.

You send command data to the interface board containing the following information

1. The Interface Board you want to address
2. Whether you want to want to send data to the board (**WRITE)** or read data back from the board (**READ**)
3. The destination of the data:- The board itself, a particular motor or a particular sensor.
4. What you want the sensor to do.

If it’s a **WRITE** Command you can command the board to:

* 1. Change default values

The ability to change set-up parameters for individual sensors is provided for Super Users. (see Appendix TBD)

b. Initialise the Stepper Motor to its zero datum (using the micro switch)

c. The angle to rotate the sensor to between limits of +/- 127 degrees

1. A specific fixed angle
2. Two angles to define the boundaries of an arc or segment to continuously scan between.
3. The speed of sensor rotation (scan) required.

d. The type of scan to undertake.

1. Sample **sensor range** values continuously for transmission of the current value and associated angle when requested.
2. Sample **sensor range** values during a single scan and determine the minimum range and associated angle for transmission when requested.
3. **Track** the object detected at minimum range during the initial scan and when requested transmit the range and associated angle of that object.

If it’s a **READ** Command you can command the board to:

1. Send back the latest **Odometer** ‘distance travelled’ readings for each wheel
2. Send back the **Range and Angle** (depending in which mode you requested the sensor to undertake)
3. **Raspberry Pi I2C Interface to Sensors Board**

The Interface between the Raspberry Pi and the Scanning motors and Sensors is by a two wire I2C synchronous serial link. An interface board is provided to handle time critical tasks that could monopolise the computing resources of the Raspberry Pi and delay the execution of the main navigation & control programme.

The R-Pi is the I2C Master and the Sensors Interface board is the I2C Slave.

**R-Pi I2C Programme Extract Written in Python**

import smbus, time

bus = smbus.SMbus(1) #There are two SMbus available on the R-Pi

address = 4 #Seven bit Byte: as bit 8 is used for **READ/WRITE** designation.

control = 1 #Tells sensor board slave what sensor to address and actions to undertake

TxByte1 = 0 #Zero TxByte1

TxByte2 = 0 #Zero TxByte2

TxBytes = [TxByte1, TxByte2] #Put TxByte1 & TxByte2 into an array

numbytes = 4 #Number of bytes to be received on a **READ** instruction

**# R-Pi I2C WRITE**

**Bus.write\_block\_data(address, control, TxBytes)**

# The R-Pi will also insert a byte between control and TxBytes containing the number of bytes

# in the TxBytes array, i.e. 2

# The number of bytes transmitted by the R-Pi is therefore 5

**# R-Pi I2C READ**

**RxBytes = bus.read\_i2c\_block\_data(address, control, numbytes)**

# The R-Pi will send: address+0(for **WRITE**), control, *restart(bus held high),* address+1(for **READ**),

# *followed by* RxByte[1], RxByte[2], RxByte[3], RxByte[4]

# The R-Pi will extend or shorten the RxByte array according to numbytes designated above.

**I2C Slave**

The Sensors Interface board I2C slave will match the address sent by the R-Pi and if it is a match

it will interrupt what it is doing and receive the I2c instructions.

If it is an **R-Pi WRITE** it will extract the data into an array of 3 bytes

containing the control byte and the 2 data bytes transmitted.

If it is an **R-Pi READ** it will reply with n bytes of data as requested by the control byte and numbytes

For the **Odometers** this will be 4 bytes containing

The high and low bytes of unscaled left wheel distance followed by

The high and low bytes of the unscaled right wheel distance. (approximate mm)

For the **Range Sensor** this will be 4 bytes containing

A moving/stationary flag followed by the high byte of the angle (approximate +/-tenths of a degree)

The low byte of the angle

The high byte of unscaled sensor distance (approximate mm)

The low byte of unscaled sensor distance

1. **WRITE Control Codes**

**Decimal Control Code TxByte1 TxByte2**

Bits[7-5] Sensor options: (decimal bit values: 128, 64, 32)

000 = Stepper Motor 0

001 = Servo motor 32

010 = Infra Red Ranging 64

011 = Ultra Sonic Ranging 96

100 = Odometers 128

101 = TBD 160

110 = TBD 192

111 = TBD 224

Bits[4-2] Types of scan: (decimal bit values: 16, 8, 4)

000 = Initialise to datum 0 0 0

001 = Fixed Angle (+/-deg) 4 Angle 0

010 = Scan between two angles (+/-deg) 8 Left Angle Right Angle

011 = TBD 12

100 = TBD 16

101 = TBD 20

110 = TBD 24

111 = EEPROM Default values 28 Address Data

Bits[1-0] Speed of Scan in deg/sec (decimal bit values: 2, 1)

00 = 0 (Stationary) 0

01 = 20 deg/sec 1

10 = 40 deg/sec 2

11 = 60 deg/sec (Maximum) 3

To obtain the control code in decimal, add the appropriate numbers (binary bit values) together.

e.g. Addressing a servo motor to scan between two angles at 40 deg/sec the Control Code = 32+8+2= 42

1. **READ Control Codes**

**Decimal Control Code RxByte1 RxByte2 RxByte3 RxByte4**

Bits[7-5] Sensor options: (decimal bit values: 128, 64, 32)

000 = Stepper Motor 0 na na

001 = Servo motor 32 na na

010 = Infra Red Ranging Current Reading 64 Moving Flag & Angle Range of obstacle

011 = Infra Red Ranging Lowest Reading 96 Moving Flag & Angle Min Range of obstacle

100 = Ultra Sonic Ranging Current Reading 128 Moving Flag & Angle Range of obstacle

101 = Ultra Sonic Ranging Lowest Reading 160 Moving Flag & Angle Min Range of obstacle

110 = Odometers 192 Left Distance Right Distance

111 = TBD 224

If it is an **R-Pi READ** it will reply with n bytes of data as requested by the control byte and numbytes

For the **Odometers** this will be 4 bytes containing

The high and low bytes of unscaled left wheel distance followed by

The high and low bytes of the unscaled right wheel distance. (approximate mm)

For the **Range Sensor** this will be 4 bytes containing

A moving/stationary flag followed by the high byte of the angle (approximate +/-tenths of a degree)

The low byte of the angle

The high byte of unscaled sensor distance (approximate mm)

The low byte of unscaled sensor distance

1. **Sensor and Data Transfer Accuracy, Limits and Timing.**

**Sensor Accuracy** depends on the sensor construction and its purchase cost reflects this. The sensors supplied are adequate, but are not high accuracy.

**Data Transfer Accuracy** has been chosen by us to exceed the accuracy of the motors and sensors.

A single ‘byte’ containing 8 bits can hold numbers from 0 to 255 or +/-127. At a resolution of 1 deg/bit the unambiguous scan limits are +/-127 degrees which requires 1 byte of data.

A ‘word’ (or 2 bytes) containing 16 bits can hold numbers from 0 to 65535 or +/- 32767.

Angles to an accuracy of 0.1 degrees, and ranges to an accuracy of 1mm require 12 bits (0 to 4096). This leaves 4 bits in a transmitted word available to be used for other data such as a moving/stationary status bit.

**Data Transfers Time** depends on the transmission speed (baud rate) and the number of bytes transmitted. It is therefore important to minimise the number of bytes to ensure minimum transmission time.

At 9600 baud (bits/sec) each UART byte takes approximately 1 msec.

At 100,000 baud each I2C byte takes 0.1 msec

Data from the odometers must be collected about 4 times per revolution to prevent ambiguity. The maximum speed of a Rampaging Chariot is about 10 MPH requiring continuous data sampling within 15 msec.

Stepper Motor Pulses must be provided at a maximum frequency of about 1 msec.

**Data Limits** can depend on the number of bits used to transfer data, the frequency of data collection or the capability of the sensor or stepper motor/servo.

Odometer Scaling will be in millimetres from the start, assuming standard wheel size. The start datum is offset to 10000 mm to allow for backward movement and the maximum unambiguous unscaled forward distance is 55535 mm. the R-Pi calculation can extended this to any distance required.

**Data Scaling**

Provision for corrections for different wheel diameter and asymmetric wheel diameter are available in the Raspberry-Pi.

The R-Pi will calculate the distance travelled by the centre of the chassis and the angle the chassis has turned through since the start. Use degrees for any display, but calculations may use radians.

To conform with general mathematical convention a positive turn angle is anticlockwise and the 0 degree datum is along the x axis.

Angle scaling will be in Degrees and Tenths

Distance scaling will be in millimetres

1. **Proposed Sensors**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sensor** | **Type** | **Measures** | **Accuracy** | **Shop** | **Price** | **Interface** |
| **Odometer** | **Magnetic Rotary Encoder AS5040 with Magnet BMN-35H (Diametrically magnetised)** | **Wheel Rotation**  **Angle** | **0.9mm** | **Farnell** | **£5.48 each (10)**  **(2 reqd + magnets)** | **5v**  **Ground**  **PWM**  **2 dig Inputs** |
| **Infra-Red Ranging** | **Sharp 2Y0A02 YK Range 200mm to 1500mm** | **Obstacle Distance**  **(Angle with Scanning)** | **Medium** | **E-Bay - Hong Kong** | **£5.20** | **5v**  **Ground**  **Analogue**  **1 Analogue**  **Input** |
| **Alternative Ultra- Sonic Ranging with standard RC servo** | **HC-SR04**  **Range 20mm to 4000mm** | **Obstacle Distance**  **(Angle with Scanning)** | **Low** | **E-Bay – Hong Kong** | **£0.99** | **5v**  **Ground**  **Trigger**  **Echo**  **1 dig Output**  **1 dig Input** |
| **Stepper Motor** | **28BYJ-48 Unipolar**  **with 64:1 gearbox & ULN 2003 Driver** | **Angle of scan** | **High with gearing** | **E-Bay – Hong Kong** | **£2.45** | **5v**  **Ground**  **Phase A**  **Phase B**  **Phase C**  **Phase D**  **Datum Sw**  **4 dig Outputs**  **1 dig Input** |
| **Two channel logic level converter** | **4-channel-Bi-Directional-Logic-Level-Shifter-Converter-3-3V-5V** |  |  | **E-Bay China** | **£1.09** |  |

1. **Raspberry Pi Software Module Organisation**

The above chart shows basic dependency between components in the model. It is an object-orientated rather than a functionally orientated approach in grouping the functions.

Each module encapsulates the functionality that would be given directly by its namesake. This allows each layer up to take a further level of abstraction, until in the Main module, at the bottom the learner user can simply write:

m = Main()  
m.arena = Assault()  
m.chariot.start()  
  
for waypoint in m.arena.waypoints:  
 m.chariot.goto(waypoint)

If, then, they set Main.py to be run when the Pi is booted, or upon some other signal, set up the chariot in an assault course, and switch it on, they should be able to watch it make its way around.

1. **Other Useful Data**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sensor** | **Type** | **Measures** | **Accuracy** | **Shop** | **Price** |
| **Laser Ranging** | **LIDAR-Lite**  **LL-905-PIN-01L**  **Range 0 to 30m @ 100Hz** | **Obstacle Distance**  **(Angle with Scanning)** | **Medium**  **+/- 25mm** | **Pacer** | **£84** |
|  |  |  |  |  |  |
| Compass | Magnetic | Heading | Low |  |  |
|  | Gyro | Heading | Drifts |  |  |
| GPS |  | Position  Calc Speed & Heading | Not Indoors  High Outdoors |  |  |
| Gyros |  | Angular velocity |  |  |  |
| Accelerometer |  | Acceleration |  |  |  |
| Servo |  | Angle of Scan |  |  |  |
| Camera |  | Image Recognition |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Gadget** | **Type** | **Function** |  |  | **Shop** | **Price** |
| Rampaging Chariot | Robot | Land vehicle to house autonomous functions |  |  | Rampaging Chariots Guild | £100 Kit |
| Start/Stop Radio Switch |  | Start and stop vehicle remotely |  |  |  |  |
| Smart Phone |  | Data communication |  |  |  |  |
| Data Logger |  | Data Analysis |  |  |  |  |
| Telemetry |  | Real Time Data Analysis |  |  |  |  |
| Video Camera |  | Analysis |  |  |  |  |
| Robot Arm |  | Pick & Place |  |  |  |  |

1. Speed of sound = 340.29 m/sec at sea level = 0.34 mm/us/2 Distance in mm = us/5.88

Level Shifter between 5v and 3.3v for !2C interface to PIC.

<https://learn.sparkfun.com/tutorials/bi-directional-logic-level-converter-hookup-guide>

<http://www.ebay.co.uk/itm/4-channel-Bi-Directional-Logic-Level-Shifter-Converter-3-3V-5V-For-Arduino-/321779851075?hash=item4aeb92df43>

<http://www.ebay.co.uk/itm/AMS-AS5035-ASSU-ENCODER-MAGNETIC-ROTARY-8BIT-16SSOP-/181722373412?hash=item2a4f7f5d24>

<http://www.ebay.co.uk/itm/GP2Y0A02YK0F-20-150CM-Infrared-Proximity-Sensor-Long-Ranges-Sharp-With-Cable-/350894116899?pt=LH_DefaultDomain_3&hash=item51b2eb5023>

<http://www.ebay.co.uk/itm/2x-Ultrasonic-Module-HC-SR04-Distance-Measuring-Transducer-Sensor-for-Arduino-UK-/181688236391?pt=LH_DefaultDomain_3&hash=item2a4d767967>

<http://www.ebay.co.uk/itm/5V-28BYJ-48-4-Phase-Stepper-Motor-with-ULN2003-Driver-Board-Arduino-Pi-Pic-AVR-/281286257686?pt=LH_DefaultDomain_3&hash=item417df7ac16>

<http://www.amazon.co.uk/gp/product/B0089JV2OM/ref=pe_1941351_80356851_em_2p_0_ti>