LSE

Laboratório de Sistemas Embebidos

Purpins Comm

Gonçalo Cabrita

Document Identifier	Purpins Comm
Project	Purpins Robot
Version	1.0
Date	February, 24, 2015
Distribution	User Manual

LSE Instituto de Sistemas e Robótica Pinhal de Marrocos, Pólo 2, 3030-290 Coimbra **T** 239 796277 **F** 239 406672 **E** lino@isr.uc.pt **W** lse.isr.uc.pt

1. Objectives

The purpose of this document is to document the binary communication protocol developed for the Purpins robot. This documentation corresponds to version 11 of the Purpins protocol. The Purpins firmware can be found here.

2. Communication Protocol

2.1 Message Construction

Each message is composed by a header that always starts with a '@' (0x40), followed by the action byte, 1 to 29 (0x01 to 0X1D), ending with the size byte, which holds the number of bytes of the data field. The data field contains any data being transmitted. Finally the message ends with a CRC byte.

START BYTE	ACTION BYTE	SIZE BYTE	DATA	CRC BYTE
0x40	0x01	0x01	0x0B	0XD4

2.2 Cyclic Redundancy Check

All messages end with a CRC byte for error checking. An 8-bit CRC CCITT algorithm is performed on the the entire message (header plus data bytes). This was chosen due to the fact that the Purpins firmware was implemented using Tivaware which is able to run the CRC out-of-the-box.

2.3 Actions

The following table contains all actions currently supported by this version of the protocol,

1	PP_ACTION_GET_VERSION	Get the firmware version.
2	PP_ACTION_DRIVE	Drive the robot by sending linear and angular velocities.
3	PP_ACTION_DRIVE_MOTORS	Drive the robot by sending left and right motor speeds.
4	PP_ACTION_DRIVE_PWM	Drive the robot by sending left and right motor PWM values.
5	PP_ACTION_GET_ODOMETRY	Get the robot's odometry.
6	PP_ACTION_GET_MOTOR_SPEEDS	Get the left and right motor speeds.
7	PP_ACTION_GET_ENCODER_PULSES	Get the left and right encoder pulses.
8	PP_ACTION_GET_IMU	Get the IMU data from the MPU9150.
9	PP_ACTION_GET_IR_SENSORS	Get the range from the five IR sensors.
10	PP_ACTION_GET_GAS_SENSOR	Get the gas sensor reading.
20	PP_ACTION_SET_SENSORS_PACK	Set a pack of sensors for retrieval.
21	PP_ACTION_GET_SENSORS_PACK	Get a pack of sensors defined with 20.
22	PP_ACTION_SET_SENSOR_STREAMING	Start/stop streaming a pack of sensors defined with 20.
23	PP_ACTION_SET_GLOBAL_POSE	Set the robot's global pose.
24	PP_ACTION_SET_NEIGHBORS_POSES	Set the global poses of the neighbour robots.
29	PP_ACTION_ERROR	Reporting an error from the robot to the host computer.

LSE

2.3.1 Get firmware version

Get the firmware version from the robot.

Reply // 1 data byte // uint8_t version

Example

uint8_t version = 11

Request	0x40 0x01 0x00 0x93
Reply	0x40 0x01 0x01 0x0B 0XD4

2.3.2 Drive the robot

Drive the robot by setting linear (m/s) and angular speeds (rad/s).

Request // 8 data bytes // float linear speed (m/s), float angular speed (rad/s)

Example

float linear_speed = 0.1 (m/s)
float angular_speed = 0.0 (rad/s)

Request	0X40 0X02 0X08 0XCD 0XCC 0XCC 0X3D 0X00 0X00 0X00 0X00 0XE3
Reply	0X40 0X02 0X00 0XAC

2.3.3 Drive the motors

Drive the robot by setting left (rad/s) and right (rad/s) motor speeds.

Request // 8 data bytes // float left speed (m/s), float right speed (rad/s)

Example

float left_speed = 3.8 (rad/s)
float right_speed = 4.1 (rad/s)

Request	0X40 0X03 0X08 0X33 0X33 0X73 0X40 0X33 0X33 0X83 0X40 0X35
Reply	0X40 0X03 0X00 0XB9

2.3.4 Drive PWM

Drive the robot by setting left and right motor PWM values.

Request // 8 data bytes // int32_t left PWM, int32_t right PWM

Example

int32_t left_pwm = 0
int32_t right_pwm = 200

Request	0X40 0X04 0X08 0X00 0X00 0X00 0X00 0XC8 0X00 0X00 0X00 0XA9
Reply	0X40 0X04 0X00 0XD2

2.3.5 Get Odometry

Get the robot's odometry, x, y and yaw values.

Reply // 12 data bytes // float x (m), float y (m), float yaw (rad)

LSE

Example

float x = 0 (m)

float y = 0 (m)

float yaw = 0 (rad)

Request	0X40 0X05 0X00 0XC7
Reply	0X40 0X05 0X0C 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X19

2.3.6 Get Motor Speeds

Get the robot's left and right motor speeds.

Reply // 12 data bytes // float left speed (rad/s), float right speed (m/s)

Example

float left_speed = 0 (m)

float right_speed = 0 (m)

Request	0X40 0X06 0X00 0XF8
Reply	0X40 0X06 0X08 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X54

2.3.7 Get Encoder Pulses

Get the robot's left and right encoder pulses.

Reply // 12 data bytes // int32_t left pulses, int32_t right pulses

Example

int32_t left_pulses = 0

int32_t right pulses = -788529152

Request	0X40 0X07 0X00 0XED
Reply	0X40 0X07 0X08 0X00 0X00 0X00 0X00 0XD1 0X00 0X00 0X00 0XD0

2.3.8 Get IMU

Get the IMU data from the MPU9150.

Reply // 40 data bytes //

Example

float quaternion_x = 0

float quaternion_y = 0

float quaternion_z = 0

float quaternion_w = 0

float quaternion_x = 0

float quaternion_y = 0

float quaternion_w = 0

float quaternion_x = 0

float quaternion_y = 0

float quaternion_w = 0

Request	0X40 0X08 0X00 0X2E
Reply	0X40 0X08 0X28 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X47

2.3.20 Set Sensors Pack

Define a pack of sensors to be later retrieved in bulk. Sensors can be any action between 5 (0x05) and 8 (0x08), up to 10 sensors can be set.

Request // n data bytes // uint8_t sensor 1, uint8_t sensor 2, …, uint8_t sensor n

Example

uint8_t sensor_1 = 5
uint8_t sensor_2 = 6

Request	0X40 0X14 0X02 0X05 0X06 0X72
Reply	0X40 0X14 0X00 0X85

2.3.21 Get Sensors Pack

Get a pack of sensors according to what was set using set sensors pack (action 20).

Reply // n data bytes // depends on the sensors being sent

Example

float x = 0 (m)
float y = 0 (m)
float yaw = 0 (rad)

float left_speed = 0 (m)
float right_speed = 0 (m)

Request	0X40 0X15 0X00 0X9
Reply	0X40 0X15 0X14 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0X00 0XD1

2.3.22 Set Sensor Streaming

Get a pack of sensors according to what was set using set sensors pack (action 20) at a frequency defined by the user.

Reply // 4 data bytes // float frequency (Hz)

Example

float streaming_frequency = 5.0 (Hz)

Request	0X40 0X16 0X04 0X00 0X00 0XA0 0X40 0X7D
Reply	0X40 0X16 0X00 0XAF

After setting a frequency different than 0, the reply of get sensors pack (action 21) will start streaming at the desired frequency. If a frequency of 0 is set the streaming stops.

2.3.23 Set Sensor Streaming

Get a pack of sensors according to what was set using set sensors pack (action 20) at a frequency defined by the user.

Reply // 4 data bytes // float frequency (Hz)

LSE

Example

float streaming_frequency = 5.0 (Hz)

Request	0X40 0X16 0X04 0X00 0X00 0XA0 0X40 0X7D
Reply	0X40 0X16 0X00 0XAF

2.3.24 Set Sensor Streaming

Get a pack of sensors according to what was set using set sensors pack (action 20) at a frequency defined by the user.

Reply // 4 data bytes // float frequency (Hz)

Example

float streaming_frequency = 5.0 (Hz)

Request	0X40 0X16 0X04 0X00 0X00 0XA0 0X40 0X7D
Reply	0X40 0X16 0X00 0XAF

2.4 Error Management

A total of 4 different errors can be sent as a reply in place of a regular reply indicating that an error occurred. An error message is defined by an action byte of 29 (0x1D).

1	PP_ERROR_UNKNOWN_ACTION	Unknown action received
2	PP_ERROR_CRC	The CRC didn't match
3	PP_ERROR_BUFFER_SIZE	The buffer exceeded its limit
4	PP_ERROR_SENSOR_NOT AVAILABLE	The requested sensor is currently offline

Example

Unknown action.

Request	0X40 0XF0 0X00 0X00
Reply	0X40 0X1D 0X01 0X01 0XBA

3. Implementation

3.1 Structure Padding

Data alignment means putting the data at a memory offset equal to some multiple of the word size, which increases the system's performance due to the way the CPU handles memory. To align the data, it may be necessary to insert some meaningless bytes between the end of the last data structure and the start of the next, which is data structure padding. Be aware of data structure padding when retrieving the data component of a message on a host computer.

Read more here.

3.2 Python Example

A set of Python examples is distributed with the firmware, next follows a couple of examples of how to drive the robot and retrieve the odometry.

3.2.1 Drive

In this example the robot is sent linear and angular velocities.

```
import serial
import crcmod.predefined
import struct
import sys

if len(sys.argv) != 3:
    print "usage: drive.py <linear_speed> <angular_speed>"
    sys.exit(2)

port = serial.Serial(port='/dev/tty.usbmodem0E20D771', baudrate=115200)

header = [0x40, 0x02, 0x08]

msg = "".join(map(chr, header))
msg += struct.pack("2f", float(sys.argv[1]), float(sys.argv[2]))

crc = crcmod.predefined.Crc('crc-8')
crc.update(msg)

msg += chr(crc.crcValue)

print "Request " + " ".join("{:#04X}".format(ord(i)) for i in msg)

port.write(msg)

reply = port.read(4)

print "Reply   " + " ".join("{:#04X}".format(ord(i)) for i in reply)

port.close()
```

3.2.2 Odometry

In this example the robot is asked for its odometry.

```
import serial
import crcmod.predefined
import struct
import sys

port = serial.Serial(port='/dev/tty.usbmodem0E20D771', baudrate=115200)

header = [0x40, 0x05, 0x00]

msg = "".join(map(chr, header))

crc = crcmod.predefined.Crc('crc-8')
crc.update(msg)

msg += chr(crc.crcValue)

print "Request " + " ".join("{:#04X}".format(ord(i)) for i in msg)

port.write(msg)

reply = port.read(16)

print "Reply   " + " ".join("{:#04X}".format(ord(i)) for i in reply)

reply_components = struct.unpack('!3B3fB', reply)

crc = crcmod.predefined.Crc('crc-8')
crc.update(reply[0:15])

if crc.crcValue != reply_components[6]:
    print "ERROR! CRC does not match!"
    sys.exit(2)

print " "
print "x   " + str(reply_components[3])
print "y   " + str(reply_components[4])
print "yaw " + str(reply_components[5])
print " "

port.close()
```