Pi-puck Documentation

University of York

Oct 15, 2020

GENERAL INFORMATION

1 About the Pi-puck **3**

2 Publications **5**
2.1 Primary Publications 5
2.2 Citations 5

3 Hardware Overview **7**
3.1 Schematic 7
3.2 Block diagram 7

4 Software Overview **9**
4.1 Software APIs 9
4.2 Code Repositories 9

5 Pi-puck Python Library **11**
5.1 Pi-puck Controllers 11
5.2 e-puck Controllers 13
5.3 Expansion Board Controllers 14
5.4 Sensor Board Controllers 15
5.5 Additional Drivers 15

6 Extensions Overview **19**
6.1 YRL Expansion Board 19
6.2 Time-of-Flight Distance Sensor 19

7 YRL Expansion Board **21**
7.1 Hardware Design 22
7.2 XBee Interface 24

8 Time-of-Flight Distance Sensor **27**

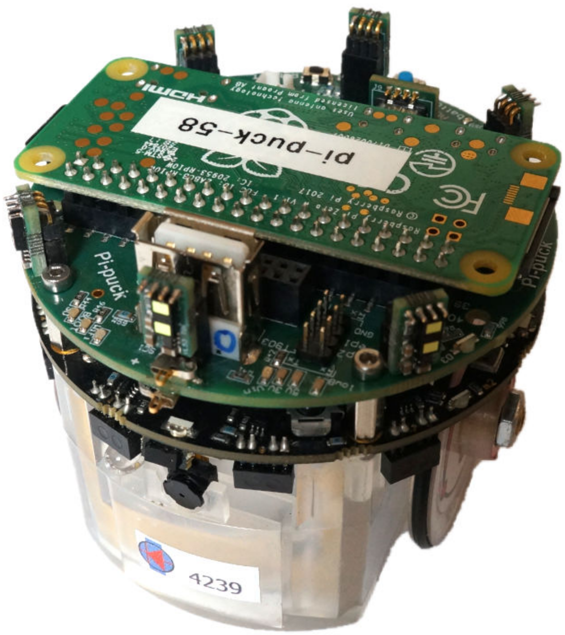
9 External Links **29**

10 Code Repositories **31**

Python Module Index **33**

Index **35**

A Raspberry Pi interface for the e-puck robot platform.
Developed jointly by the York Robotics Laboratory (YRL) at the University of York, and GCtronic.



The Pi-puck is an extension for the e-puck and e-puck2, allowing a Raspberry Pi Zero single-board computer to be mounted on the robot, adding Linux support, additional peripherals, and further expansion possibilities.

Documentation on this website is currently a work in progress.

CHAPTER

ONE

ABOUT THE PI-PUCK

The Pi-puck is an extension for the e-puck and e-puck2 robot platforms, allowing a Raspberry Pi Zero W[1] to be mounted on the robot, adding Linux support, additional peripherals, and further expansion possibilities.

It was developed as a joint project between the York Robotics Laboratory (YRL) at the University of York, and GCtronic.

[1] Also compatible with Raspberry Pi Zero WH (with headers included), Raspberry Pi Zero (without built-in wireless), or potentially other Raspberry Pi compatible boards.

CHAPTER

TWO

PUBLICATIONS

The following is a list of known academic publications related to the Pi-puck.

2.1 Primary Publications

The Pi-puck extension board: a Raspberry Pi interface for the e-puck robot platform
Alan G. Millard, Russell Joyce, James A. Hilder, Cristian Fleşeriu, Leonard Newbrook, Wei Li, Liam J. McDaid, and David M. Halliday
2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2017), Sep. 2017
https://doi.org/10.1109/IROS.2017.8202233
https://eprints.whiterose.ac.uk/120310/

2.2 Citations

A distributed vision-based navigation system for Khepera IV mobile robots
Gonzalo Farias, Ernesto Fabregas, Enrique Torres, Gaëtan Bricas, Sebastián Dormido-Canto, and Sebastián Dormido
Sensors, volume 20, article 5409, Sep. 2020
https://doi.org/10.3390/s20185409

Human-swarm interaction via e-ink displays
Alan G. Millard, Russell Joyce, and Ian Gray
ICRA 2020 Human-Swarm Interaction Workshop, May 2020
https://eprints.whiterose.ac.uk/161398/

A blockchain-controlled physical robot swarm communicating via an ad-hoc network
Alexandre Pacheco, Volker Strobel, and Marco Dorigo
IRIDIA – Technical Report Series, number TR/IRIDIA/2020-007, May 2020
https://iridia.ulb.ac.be/IridiaTrSeries/link/IridiaTr2020-007.pdf

HuGoS: a multi-user virtual environment for studying human–human swarm intelligence
Nicolas Coucke, Mary K. Heinrich, Axel Cleeremans, and Marco Dorigo
IRIDIA – Technical Report Series, number TR/IRIDIA/2020-003, May 2020
https://iridia.ulb.ac.be/IridiaTrSeries/link/IridiaTr2020-003.pdf

Classification and management of computational resources of robotic swarms and the overcoming of their constraints
Stefan M. Trenkwalder
PhD thesis, University of Sheffield, Mar. 2020
https://etheses.whiterose.ac.uk/26510/

A framework for swarm robotics experimentation with Pi-puck robots and an Ethereum-based blockchain
Alexandre Pacheco, Volker Strobel, and Marco Dorigo
IRIDIA – Technical Report Series, number TR/IRIDIA/2020-001, Feb. 2020
https://iridia.ulb.ac.be/IridiaTrSeries/link/IridiaTr2020-001.pdf

Blockchain technology secures robot swarms: a comparison of consensus protocols and their resilience to Byzantine robots
Volker Strobel, Eduardo Castelló Ferrer, and Marco Dorigo
IRIDIA – Technical Report Series, number TR/IRIDIA/2019-004, Nov. 2019
https://iridia.ulb.ac.be/IridiaTrSeries/link/IridiaTr2019-004.pdf

BulbRobot – inexpensive open hardware and software robot featuring catadioptric vision and virtual sonars
João Ferreira, Filipe Coelho, Armando Sousa, and Luís Paulo Reis
Fourth Iberian Robotics Conference (ROBOT 2019), Nov. 2019
https://doi.org/10.1007/978-3-030-35990-4_45

SwarmCom: an infra-red-based mobile ad-hoc network for severely constrained robots
Stefan M. Trenkwalder, Iñaki Esnaola, Yuri Kaszubowski Lopes, Andreas Kolling, and Roderich Groß
Autonomous Robots, volume 44, pages 93-114, Aug. 2019
https://doi.org/10.1007/s10514-019-09873-0

Command language for single-user, multi-robot swarm control
Abraham M. Shultz
PhD dissertation, University of Massachusetts, Dec. 2018
https://search.proquest.com/openview/af784d65959b38c2ba0c537071066d1a/1

ARDebug: An augmented reality tool for analysing and debugging swarm robotic systems
Alan G. Millard, Richard Redpath, Alistair M. Jewers, Charlotte Arndt, Russell Joyce, James A. Hilder, Liam J. McDaid, and David M. Halliday
Frontiers in Robotics and AI, volume 5, article 87, Jul. 2018
https://doi.org/10.3389/frobt.2018.00087

A two teraflop swarm
Simon Jones, Matthew Studley, Sabine Hauert, and Alan F. T. Winfield
Frontiers in Robotics and AI, volume 5, article 11, Feb. 2018
https://doi.org/10.3389/frobt.2018.00011

CHAPTER

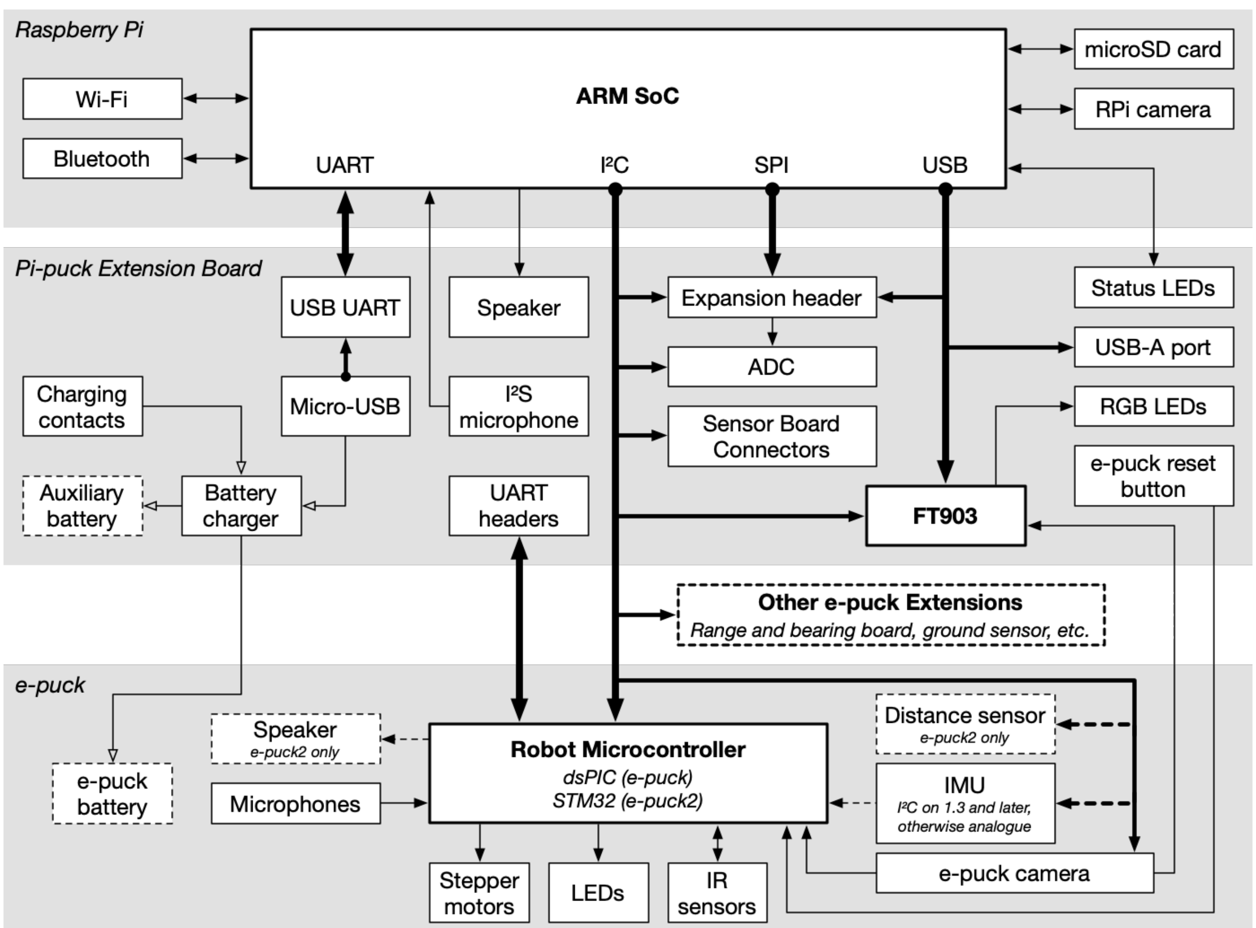
THREE

HARDWARE OVERVIEW

3.1 Schematic

`Pi-puck board schematic` by GCtronic.

3.2 Block diagram



`Download block diagram as PDF`

CHAPTER

FOUR

SOFTWARE OVERVIEW

The YRL software distribution for the Pi-puck includes a custom Raspbian image, several Debian packages enabling full support for the hardware, and a number of additional utilities and libraries for controlling the robot.

4.1 Software APIs

- *Pi-puck Python Library*

4.2 Code Repositories

- YRL Pi-puck main repository
- YRL Pi-puck e-puck1 dsPIC firmware
- YRL Pi-puck FT903 firmware
- YRL Pi-puck Debian packages
- YRL Pi-puck Raspbian distribution
- YRL Pi-puck ROS driver package

CHAPTER

FIVE

PI-PUCK PYTHON LIBRARY

Python library for controlling the Pi-puck.

5.1 Pi-puck Controllers

pipuck.pipuck.PiPuck(epuck_version, ...)	Main Pi-puck controller class.
pipuck.ft903.FT903(bus, i2c_address)	Class to interface with the FT903 microcontroller.

5.1.1 Pi-puck Controller

Python module for controlling the Pi-puck.

class pipuck.pipuck.**PiPuck**(*epuck_version: Optional[int] = None*, *tof_sensors: Sequence[bool] = False, False, False, False, False, False*, *yrl_expansion: bool = False*)

Main Pi-puck controller class.

Parameters

- **epuck_version** – version of the base e-puck robot (either 1 or 2)
- **tof_sensors** – time-of-flight sensor boards attached (6 element tuple/list of `bool` values)
- **yrl_expansion** – YRL Expansion Board attached

property battery_is_charging

Whether the robot is connected to a charger (either through USB or the charging contacts).

Returns `True` if charging, otherwise `False`

static convert_adc_to_voltage(*adc_value: str*, *scale: float = 1.0*) → float

Convert ADC reading to voltage.

Parameters

- **adc_value** – raw value from ADC file (in mV)
- **scale** – scaling factor of the ADC channel

Returns current measured battery voltage (in V)

epuck

e-puck robot controller, either `None` or instance of *pipuck.epuck.EPuck*

expansion
expansion board controller, either `None` or instance of *`pipuck.yrl_expansion.YRLExpansion`*

ft903
FT903 microcontroller controller, instance of *`pipuck.ft903.FT903`*

get_battery_state(*battery_type: str = 'epuck'*) → Tuple[bool, float, float]
Get current battery state.

Parameters **battery_type** – battery type to check (either 'epuck' or 'aux')

Returns tuple of (charging state, battery voltage, approximate battery percentage)

set_led_colour(*led, colour: str*) → None
Set a single RGB LED colour.

Parameters

- **led** – the LED to set (0-2)
- **colour** – colour value (off/black/red/green/yellow/blue/magenta/cyan/white)

set_led_raw(*led: int, value: int*) → None
Set a single RGB LED colour.

Parameters

- **led** – the LED to set (0-2)
- **value** – raw data byte value (`0b00000BGR`)

set_led_rgb(*led: int, red: bool, green: bool, blue: bool*) → None
Set a single RGB LED colour.

Parameters

- **led** – the LED to set (0-2)
- **red** – red value (on/off)
- **green** – green value (on/off)
- **blue** – blue value (on/off)

set_leds_colour(*colour: str*) → None
Set all RGB LEDs to the same colour.

Parameters **colour** – colour value (off/black/red/green/yellow/blue/magenta/cyan/white)

set_leds_colours(*colour1: str, colour2: str, colour3: str*) → None
Set RGB LEDs to the specified colours.

Parameters

- **colour1** – LED1 colour (off/black/red/green/yellow/blue/magenta/cyan/white)
- **colour2** – LED2 colour (off/black/red/green/yellow/blue/magenta/cyan/white)
- **colour3** – LED3 colour (off/black/red/green/yellow/blue/magenta/cyan/white)

set_leds_rgb(*red: bool, green: bool, blue: bool*) → None
Set all RGB LEDs to the same colour.

Parameters

- **red** – red value (on/off)
- **green** – green value (on/off)

- **blue** – blue value (on/off)

static speaker_enable(*state: bool*) → None
Enable or disable the Pi-puck speaker audio amplifier.

Parameters state – `True` to enable or `False` to disable

tof_sensors
time-of-flight sensor controllers, 6-tuple of either *pipuck.tof_sensor.ToFSensor* instances or `None`

5.1.2 FT903 Controller

class pipuck.ft903.**FT903**(*bus: smbus.SMBus, i2c_address: int = 28*)
Class to interface with the FT903 microcontroller.

Parameters

- **bus** – `smbus.SMBus` instance to use for communication
- **i2c_address** – I2C slave address of the FT903 chip (default `0x1C`)

read_data_16(*address*)

read_data_8(*address*)

write_data_16(*address, data*)

write_data_8(*address, data*)

5.2 e-puck Controllers

pipuck.epuck.EPuck(i2c_bus, i2c_address)	Class for interfacing with a generic e-puck robot.
pipuck.epuck1.EPuck1(i2c_bus, i2c_address)	Class for interfacing with an e-puck1 robot
pipuck.epuck2.EPuck2(i2c_bus, i2c_address)	Class for interfacing with an e-puck2 robot

5.2.1 Generic e-puck controller

class pipuck.epuck.**EPuck**(*i2c_bus: Optional[int] = None, i2c_address: Optional[int] = None*)
Class for interfacing with a generic e-puck robot.

static reset_robot()

5.2.2 e-puck1 controller

class pipuck.epuck1.**EPuck1**(*i2c_bus: Optional[int] = None, i2c_address: Optional[int] = None*)
Bases: *pipuck.epuck.EPuck*

Class for interfacing with an e-puck1 robot

enable_ir_sensors(*enabled*)

get_ir_ambient(*sensor*)

get_ir_reflected(*sensor*)

property ir_ambient

property ir_reflected

property left_motor_speed

property left_motor_steps

property motor_speeds

property motor_steps

static reset_robot()

property right_motor_speed

property right_motor_steps

set_inner_leds(*front*, *body*)

set_left_motor_speed(*speed*)

set_motor_speeds(*speed_left*, *speed_right*)

set_outer_leds(*led0*, *led1*, *led2*, *led3*, *led4*, *led5*, *led6*, *led7*)

set_outer_leds_byte(*leds*)

set_right_motor_speed(*speed*)

5.2.3 e-puck2 controller

class pipuck.epuck2.**EPuck2**(*i2c_bus: Optional[int] = None*, *i2c_address: Optional[int] = None*)

Bases: `pipuck.epuck.EPuck`

Class for interfacing with an e-puck2 robot

static reset_robot()

5.3 Expansion Board Controllers

`pipuck.yrl_expansion.YRLExpansion`(bus)	

5.3.1 YRL Expansion Board

class pipuck.yrl_expansion.**YRLExpansion**(*bus*)

get_nav_direction()

set_led_colour(*colour*)

set_led_rgb(*red*, *green*, *blue*)

5.4 Sensor Board Controllers

pipuck.tof_sensor.ToFSensor(*args, **kwargs)	

5.4.1 Time-of-Flight Sensor

class pipuck.tof_sensor.**ToFSensor**(**args: Any*, ***kwargs: Any*)

Bases: VL53L1X.VL53L1X

5.5 Additional Drivers

pipuck.mcp23017.MCP23017(i2c_bus[, address])	Supports MCP23017 instance on specified I2C bus and optionally at the specified I2C address.
pipuck.lsm9ds1.LSM9DS1(i2c_bus[, …])	Driver for the LSM9DS1 accelerometer, magnetometer, gyroscope.
VL53L1X.VL53L1X	

5.5.1 MCP23017 I/O Expander Driver

Python module for the MCP23017 I2C I/O extender.

Based on https://github.com/adafruit/Adafruit_CircuitPython_MCP230xx.git

class pipuck.mcp23017.**DigitalInOut**(*pin_number*, *mcp230xx*)

Digital input/output of the MCP230xx. The interface is exactly the same as the digitalio.DigitalInOut class (however the MCP230xx does not support pull-down resistors and an exception will be thrown attempting to set one).

Specify the pin number of the MCP230xx (0…7 for MCP23008, or 0…15 for MCP23017) and MCP23008 instance.

property direction

The direction of the pin, either True for an input or False for an output.

property pull

Enable or disable internal pull-up resistors for this pin. A value of _MCP23017_PULL_UP will enable a pull-up resistor, and None will disable it. Pull-down resistors are NOT supported!

switch_to_input(*pull=None*)

Switch the pin state to a digital input with the provided starting pull-up resistor state (optional, no pull-up by default). Note that pull-down resistors are NOT supported!

switch_to_output(*value=False*)

Switch the pin state to a digital output with the provided starting value (True/False for high or low, default is False/low).

property value

The value of the pin, either True for high or False for low. Note you must configure as an output or input appropriately before reading and writing this value.

class pipuck.mcp23017.**MCP23017**(*i2c_bus*, *address=32*)
Supports MCP23017 instance on specified I2C bus and optionally at the specified I2C address.

clear_inta()
Clears port A interrupts.

clear_intb()
Clears port B interrupts.

clear_ints()
Clears interrupts by reading INTCAP.

property default_value
The raw DEFVAL interrupt control register. The default comparison value is configured in the DEFVAL register. If enabled (via GPINTEN and INTCON) to compare against the DEFVAL register, an opposite value on the associated pin will cause an interrupt to occur.

get_pin(*pin*)
Convenience function to create an instance of the DigitalInOut class pointing at the specified pin of this MCP23017 device.

property gpio
The raw GPIO output register. Each bit represents the output value of the associated pin (0 = low, 1 = high), assuming that pin has been configured as an output previously.

property gpioa
The raw GPIO A output register. Each bit represents the output value of the associated pin (0 = low, 1 = high), assuming that pin has been configured as an output previously.

property gpiob
The raw GPIO B output register. Each bit represents the output value of the associated pin (0 = low, 1 = high), assuming that pin has been configured as an output previously.

property gppu
The raw GPPU pull-up register. Each bit represents if a pull-up is enabled on the specified pin (1 = pull-up enabled, 0 = pull-up disabled). Note pull-down resistors are NOT supported!

property gppua
The raw GPPU A pull-up register. Each bit represents if a pull-up is enabled on the specified pin (1 = pull-up enabled, 0 = pull-up disabled). Note pull-down resistors are NOT supported!

property gppub
The raw GPPU B pull-up register. Each bit represents if a pull-up is enabled on the specified pin (1 = pull-up enabled, 0 = pull-up disabled). Note pull-down resistors are NOT supported!

property int_flag
Returns a list with the pin numbers that caused an interrupt port A —-> pins 0-7 port B —-> pins 8-15

property int_flaga
Returns a list of pin numbers that caused an interrupt in port A pins: 0-7

property int_flagb
Returns a list of pin numbers that caused an interrupt in port B pins: 8-15

property interrupt_configuration
The raw INTCON interrupt control register. The INTCON register controls how the associated pin value is compared for the interrupt-on-change feature. If a bit is set, the corresponding I/O pin is compared against the associated bit in the DEFVAL register. If a bit value is clear, the corresponding I/O pin is compared against the previous value.

property interrupt_enable
The raw GPINTEN interrupt control register. The GPINTEN register controls the interrupt-on-change

feature for each pin. If a bit is set, the corresponding pin is enabled for interrupt-on-change. The DEFVAL and INTCON registers must also be configured if any pins are enabled for interrupt-on-change.

property io_control
The raw IOCON configuration register. Bit 1 controls interrupt polarity (1 = active-high, 0 = active-low). Bit 2 is whether irq pin is open drain (1 = open drain, 0 = push-pull). Bit 3 is unused. Bit 4 is whether SDA slew rate is enabled (1 = yes). Bit 5 is if I2C address pointer auto-increments (1 = no). Bit 6 is whether interrupt pins are internally connected (1 = yes). Bit 7 is whether registers are all in one bank (1 = no).

property iodir
The raw IODIR direction register. Each bit represents direction of a pin, either 1 for an input or 0 for an output mode.

property iodira
The raw IODIR A direction register. Each bit represents direction of a pin, either 1 for an input or 0 for an output mode.

property iodirb
The raw IODIR B direction register. Each bit represents direction of a pin, either 1 for an input or 0 for an output mode.

5.5.2 LSM9DS1 IMU Driver

Sensor driver for the LSM9DS1 IMU.

Based on https://github.com/adafruit/Adafruit_CircuitPython_LSM9DS1 with patches to use SMBus and to fix some minor inconsistencies compared to the datasheet.

class `pipuck.lsm9ds1.`**`LSM9DS1`**(*i2c_bus*, *mag_address=30*, *xg_address=107*)
Driver for the LSM9DS1 accelerometer, magnetometer, gyroscope.

Initialise the LSM9DS1.

property accel_range
Accelerometer range.

Must be a value of:

- ACCELRANGE_2G
- ACCELRANGE_4G
- ACCELRANGE_8G
- ACCELRANGE_16G

property acceleration
Accelerometer X, Y, Z axis values as a 3-tuple of m/s^2 values.

close()
Close the I2C bus.

property gyro
Gyroscope X, Y, Z axis values as a 3-tuple of degrees/second values.

property gyro_scale
Gyroscope scale.

Must be a value of:

- GYROSCALE_245DPS
- GYROSCALE_500DPS

- GYROSCALE_2000DPS

property mag_gain

Magnetometer gain.

Must be a value of:

- MAGGAIN_4GAUSS
- MAGGAIN_8GAUSS
- MAGGAIN_12GAUSS
- MAGGAIN_16GAUSS

property magnetic

Magnetometer X, Y, Z axis values as a 3-tuple of gauss values.

read_accel_raw()

Read raw accelerometer sensor and return it as a 3-tuple of X, Y, Z values that are 16-bit unsigned values.

If you want the acceleration in nice units you probably want to use the accelerometer property!

read_gyro_raw()

Read raw gyroscope sensor values and return as a 3-tuple of X, Y, Z values that are 16-bit unsigned values.

If you want the gyroscope in nice units you probably want to use the gyroscope property!

read_mag_raw()

Read raw magnetometer sensor and return as a 3-tuple of X, Y, Z values that are 16-bit unsigned values.

If you want the magnetometer in nice units you probably want to use the magnetometer property!

read_temp_raw()

Read the raw temperature sensor value and return it as a 12-bit signed value.

If you want the temperature in nice units you probably want to use the temperature property!

property temperature

Temperature of the sensor in degrees Celsius.

5.5.3 VL53L1X ToF Sensor Driver

VL53L1X.**VL53L1X**

See https://github.com/pimoroni/vl53l1x-python/tree/multiple-i2c-bus-support.

CHAPTER

SIX

EXTENSIONS OVERVIEW

The Pi-puck has two main interfaces for extensions – the 20-pin multi-interface auxiliary connector (X1), and the six 4-pin I2C sensor connectors (SC1-SC6). This section documents various available extensions using those interfaces.

6.1 YRL Expansion Board

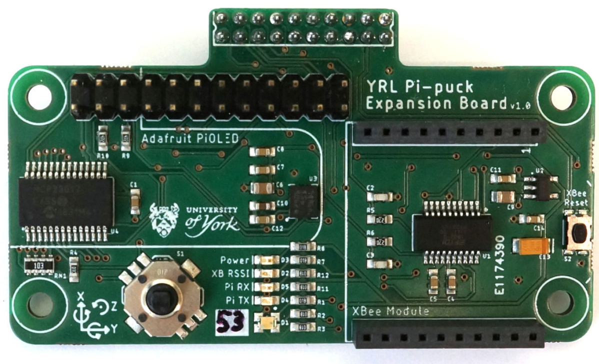
A multi-feature expansion board for the Pi-puck, developed by York Robotics Lab, that connects to SPI, I2C and USB buses through the auxiliary interface.

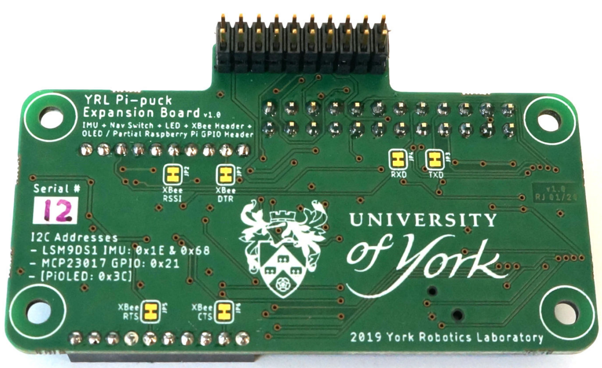
6.2 Time-of-Flight Distance Sensor

A small VL53L1X-based distance sensor add-on for the six sensor board sockets on the Pi-puck.

CHAPTER SEVEN

YRL EXPANSION BOARD





A multi-feature expansion board for the Pi-puck, developed by York Robotics Lab, that connects to SPI, I2C and USB buses through the auxiliary interface.

The YRL Expansion Board has the following features:

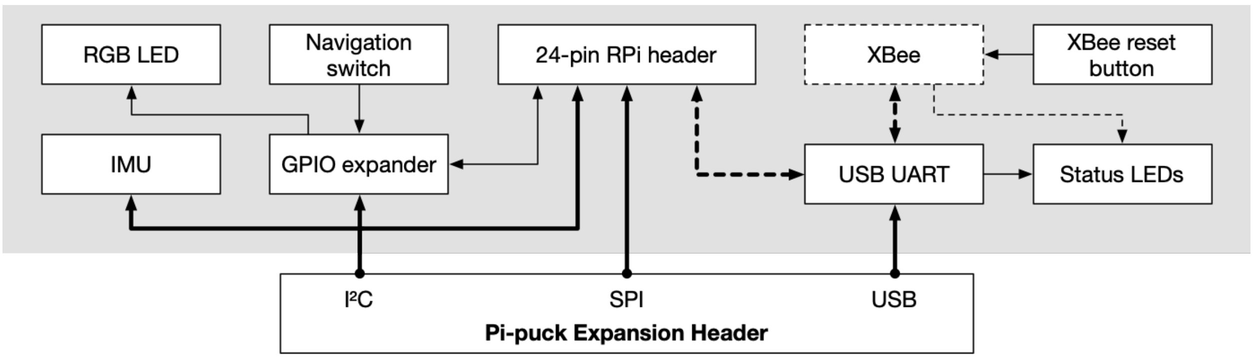
- 9-DoF LSM9DS1 IMU (accelerometer, magnetometer and gyroscope)
- XBee socket and USB UART interface
- 5-input navigation switch (up, down, left right, centre)
- RGB LED
- 24-pin Raspberry Pi compatible pin header

See https://github.com/yorkrobotlab/pi-puck-expansion-board for hardware and software sources.

7.1 Hardware Design

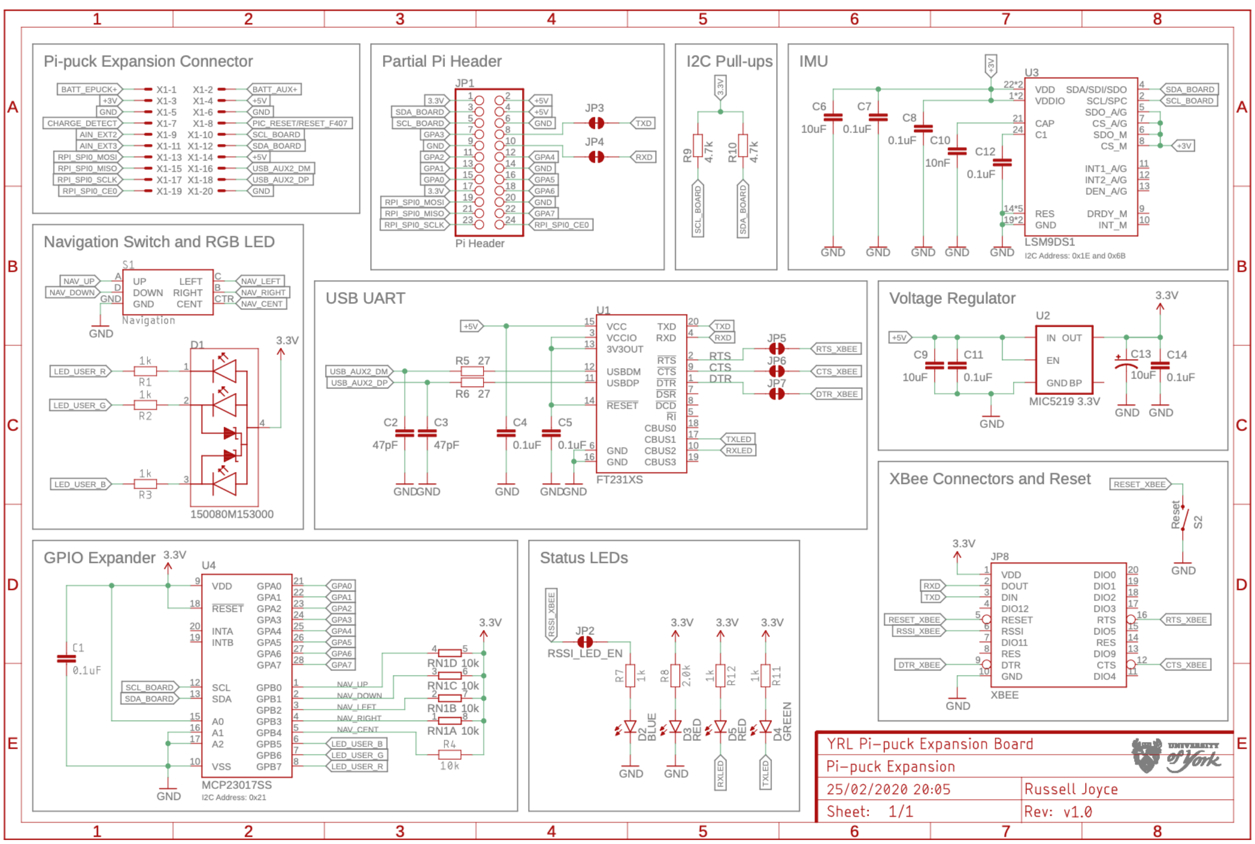
The full hardware design files in Autodesk EAGLE format are available on GitHub at https://github.com/yorkrobotlab/pi-puck-expansion-board.

7.1.1 Hardware block diagram



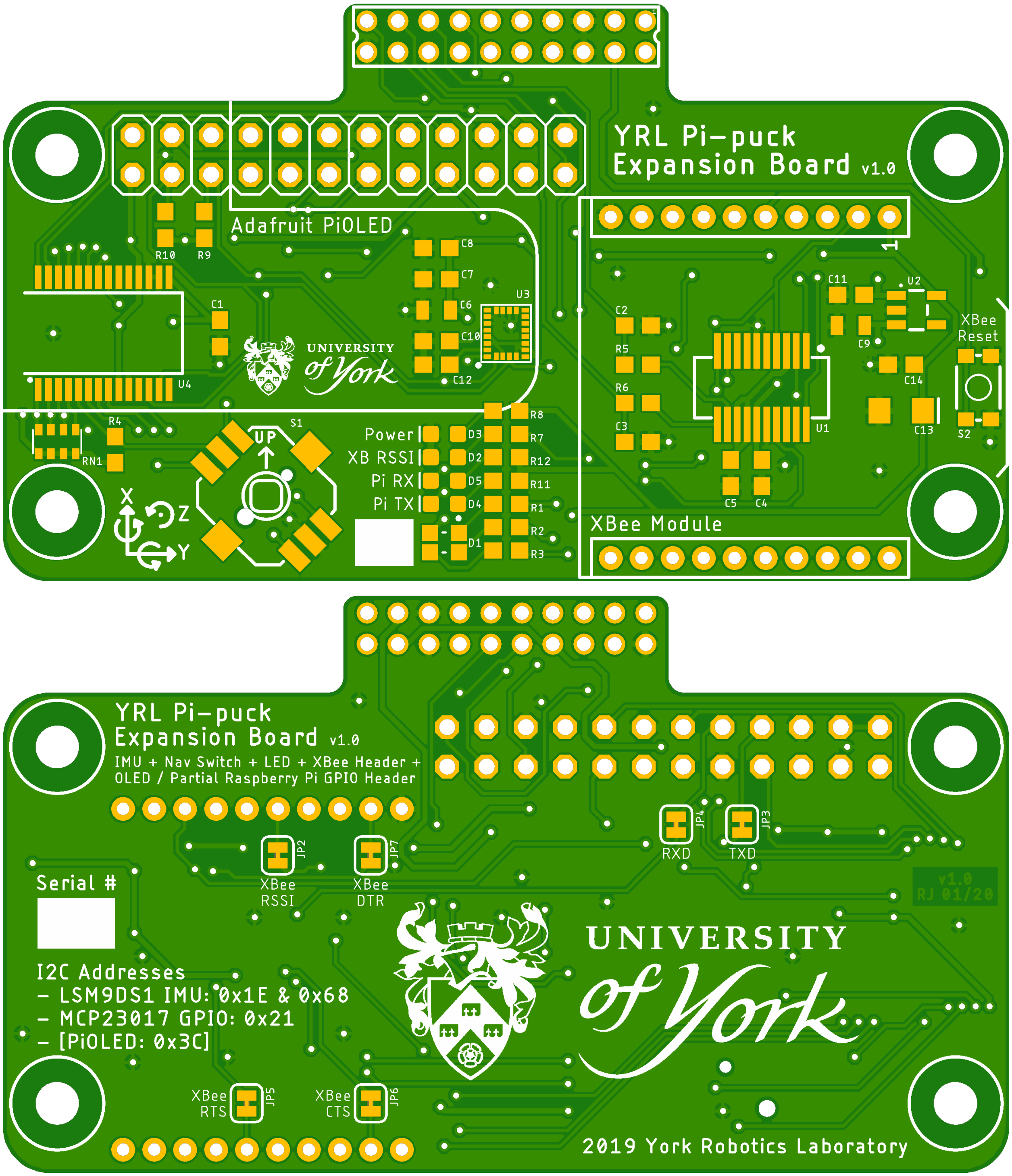
Download block diagram as PDF

7.1.2 Schematic

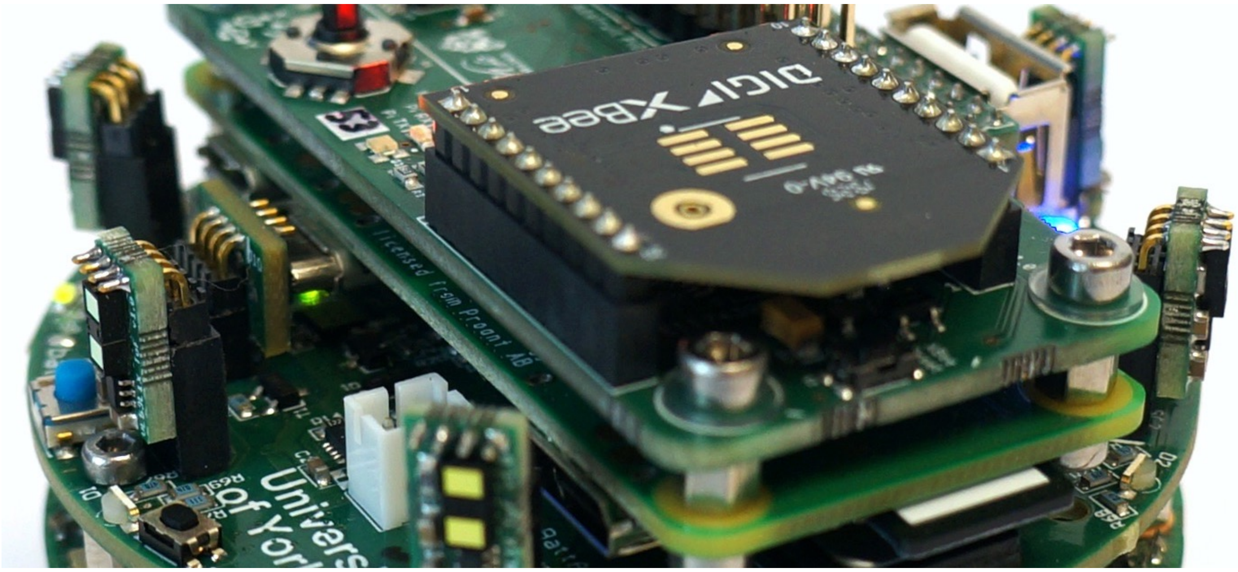


Download schematic as PDF

7.1.3 PCB layout



7.2 XBee Interface



The YRL Expansion Board has an interface for Digi XBee modules, allowing various additional forms of wireless communication between robots, such as ad hoc point-to-point or mesh networking.

The interface pinout is compatible with any through-hole XBee or XBee-PRO module, but has primarily been tested with the 2.4GHz/Zigbee 3.0 XBee 3 with PCB antenna. A good source of information on the various available XBee modules is the SparkFun XBee Buying Guide.

7.2.1 Using the XBee

The XBee is accessible using a USB UART interface from Linux on the Raspberry Pi, typically mapped to `/dev/ttyUSB0`. The USB device uses port 3 on bus 1, and has ID `0403:6015`, as shown by the following output from `lsusb`.

```
Bus 001 Device 004: ID 0403:6015 Future Technology Devices International, Ltd␣
↪Bridge(I2C/SPI/UART/FIFO)
Bus 001 Device 003: ID 0403:0fd8 Future Technology Devices International, Ltd
Bus 001 Device 002: ID 0424:2513 Standard Microsystems Corp. 2.0 Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

```
/:  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=dwc_otg/1p, 480M
    |__ Port 1: Dev 2, If 0, Class=Hub, Driver=hub/3p, 480M
        |__ Port 1: Dev 3, If 1, Class=Video, Driver=uvcvideo, 480M
        |__ Port 1: Dev 3, If 0, Class=Video, Driver=uvcvideo, 480M
        |__ Port 3: Dev 4, If 0, Class=Vendor Specific Class, Driver=ftdi_sio, 12M
```

While the Raspberry Pi does not support the standard Digi XCTU configuration utility for XBee hardware, AT commands can be sent manually over the USB UART interface, and programming libraries are available for a number of languages:

- Digi XBee Python library - https://github.com/digidotcom/xbee-python
- Digi XBee C library https://github.com/digidotcom/xbee_ansic_library
- Digi XBee Java library - https://github.com/digidotcom/xbee-java

- Digi XBee C# library - https://github.com/digidotcom/xbee-csharp

The YRL Expansion Board GitHub repository contains a number of example scripts for setting up and programming the firmware of the XBee using Python.

7.2.2 Hardware Design

The XBee interface hardware is based on the SparkFun XBee Explorer USB design, including an FT231X USB UART controller for communication with the Raspberry Pi, and LEDs for indicating data transfer on both the UART and wireless sides. Similar to the XBee Explorer, marked exposed traces can be cut (or rejoined) on the underside of the PCB for isolating RSSI, DTR, RTS and CTS signals if required.

CHAPTER

EIGHT

TIME-OF-FLIGHT DISTANCE SENSOR

A small VL53L1X-based distance sensor add-on for the six sensor board sockets on the Pi-puck.

Software control of the sensor can be achieved using the multiple-i2c-bus-support branch of the vl53l1x-python library from Pimoroni.

See https://github.com/yorkrobotlab/pi-puck-tof-sensor for more information.

CHAPTER

NINE

EXTERNAL LINKS

- YRL Pi-puck website
- GCtronic Pi-puck wiki
- GCtronic shop
- Pi-puck Debian package repository

CHAPTER

TEN

CODE REPOSITORIES

- YRL Pi-puck main repository
- YRL Pi-puck e-puck1 dsPIC firmware
- YRL Pi-puck FT903 firmware
- YRL Pi-puck Debian packages
- YRL Pi-puck Raspbian distribution
- YRL Pi-puck ROS driver package
- GCtronic Pi-puck repository

PYTHON MODULE INDEX

p

pipuck.epuck, 13
pipuck.epuck1, 13
pipuck.epuck2, 14
pipuck.ft903, 13
pipuck.lsm9ds1, 17
pipuck.mcp23017, 15
pipuck.pipuck, 11
pipuck.yrl_expansion, 14

INDEX

A

accel_range() (*pipuck.lsm9ds1.LSM9DS1 property*), 17
acceleration() (*pipuck.lsm9ds1.LSM9DS1 property*), 17

B

battery_is_charging() (*pipuck.pipuck.PiPuck property*), 11

C

clear_inta() (*pipuck.mcp23017.MCP23017 method*), 16
clear_intb() (*pipuck.mcp23017.MCP23017 method*), 16
clear_ints() (*pipuck.mcp23017.MCP23017 method*), 16
close() (*pipuck.lsm9ds1.LSM9DS1 method*), 17
convert_adc_to_voltage() (*pipuck.pipuck.PiPuck static method*), 11

D

default_value() (*pipuck.mcp23017.MCP23017 property*), 16
DigitalInOut (*class in pipuck.mcp23017*), 15
direction() (*pipuck.mcp23017.DigitalInOut property*), 15

E

enable_ir_sensors() (*pipuck.epuck1.EPuck1 method*), 13
EPuck (*class in pipuck.epuck*), 13
epuck (*pipuck.pipuck.PiPuck attribute*), 11
EPuck1 (*class in pipuck.epuck1*), 13
EPuck2 (*class in pipuck.epuck2*), 14
expansion (*pipuck.pipuck.PiPuck attribute*), 11

F

FT903 (*class in pipuck.ft903*), 13
ft903 (*pipuck.pipuck.PiPuck attribute*), 12

G

get_battery_state() (*pipuck.pipuck.PiPuck method*), 12
get_ir_ambient() (*pipuck.epuck1.EPuck1 method*), 13
get_ir_reflected() (*pipuck.epuck1.EPuck1 method*), 13
get_nav_direction() (*pipuck.yrl_expansion.YRLExpansion method*), 14
get_pin() (*pipuck.mcp23017.MCP23017 method*), 16
gpio() (*pipuck.mcp23017.MCP23017 property*), 16
gpioa() (*pipuck.mcp23017.MCP23017 property*), 16
gpiob() (*pipuck.mcp23017.MCP23017 property*), 16
gppu() (*pipuck.mcp23017.MCP23017 property*), 16
gppua() (*pipuck.mcp23017.MCP23017 property*), 16
gppub() (*pipuck.mcp23017.MCP23017 property*), 16
gyro() (*pipuck.lsm9ds1.LSM9DS1 property*), 17
gyro_scale() (*pipuck.lsm9ds1.LSM9DS1 property*), 17

I

int_flag() (*pipuck.mcp23017.MCP23017 property*), 16
int_flaga() (*pipuck.mcp23017.MCP23017 property*), 16
int_flagb() (*pipuck.mcp23017.MCP23017 property*), 16
interrupt_configuration() (*pipuck.mcp23017.MCP23017 property*), 16
interrupt_enable() (*pipuck.mcp23017.MCP23017 property*), 16
io_control() (*pipuck.mcp23017.MCP23017 property*), 17
iodir() (*pipuck.mcp23017.MCP23017 property*), 17
iodira() (*pipuck.mcp23017.MCP23017 property*), 17
iodirb() (*pipuck.mcp23017.MCP23017 property*), 17
ir_ambient() (*pipuck.epuck1.EPuck1 property*), 13
ir_reflected() (*pipuck.epuck1.EPuck1 property*), 13

L

left_motor_speed() (*pipuck.epuck1.EPuck1 property*), 14
left_motor_steps() (*pipuck.epuck1.EPuck1 property*), 14
LSM9DS1 (*class in pipuck.lsm9ds1*), 17

M

mag_gain() (*pipuck.lsm9ds1.LSM9DS1 property*), 18
magnetic() (*pipuck.lsm9ds1.LSM9DS1 property*), 18
MCP23017 (*class in pipuck.mcp23017*), 15
module
  pipuck.epuck, 13
  pipuck.epuck1, 13
  pipuck.epuck2, 14
  pipuck.ft903, 13
  pipuck.lsm9ds1, 17
  pipuck.mcp23017, 15
  pipuck.pipuck, 11
  pipuck.yrl_expansion, 14
motor_speeds() (*pipuck.epuck1.EPuck1 property*), 14
motor_steps() (*pipuck.epuck1.EPuck1 property*), 14

P

PiPuck (*class in pipuck.pipuck*), 11
pipuck.epuck
  module, 13
pipuck.epuck1
  module, 13
pipuck.epuck2
  module, 14
pipuck.ft903
  module, 13
pipuck.lsm9ds1
  module, 17
pipuck.mcp23017
  module, 15
pipuck.pipuck
  module, 11
pipuck.yrl_expansion
  module, 14
pull() (*pipuck.mcp23017.DigitalInOut property*), 15

R

read_accel_raw() (*pipuck.lsm9ds1.LSM9DS1 method*), 18
read_data_16() (*pipuck.ft903.FT903 method*), 13
read_data_8() (*pipuck.ft903.FT903 method*), 13
read_gyro_raw() (*pipuck.lsm9ds1.LSM9DS1 method*), 18
read_mag_raw() (*pipuck.lsm9ds1.LSM9DS1 method*), 18
read_temp_raw() (*pipuck.lsm9ds1.LSM9DS1 method*), 18
reset_robot() (*pipuck.epuck.EPuck static method*), 13
reset_robot() (*pipuck.epuck1.EPuck1 static method*), 14
reset_robot() (*pipuck.epuck2.EPuck2 static method*), 14
right_motor_speed() (*pipuck.epuck1.EPuck1 property*), 14
right_motor_steps() (*pipuck.epuck1.EPuck1 property*), 14

S

set_inner_leds() (*pipuck.epuck1.EPuck1 method*), 14
set_led_colour() (*pipuck.pipuck.PiPuck method*), 12
set_led_colour() (*pipuck.yrl_expansion.YRLExpansion method*), 14
set_led_raw() (*pipuck.pipuck.PiPuck method*), 12
set_led_rgb() (*pipuck.pipuck.PiPuck method*), 12
set_led_rgb() (*pipuck.yrl_expansion.YRLExpansion method*), 14
set_leds_colour() (*pipuck.pipuck.PiPuck method*), 12
set_leds_colours() (*pipuck.pipuck.PiPuck method*), 12
set_leds_rgb() (*pipuck.pipuck.PiPuck method*), 12
set_left_motor_speed() (*pipuck.epuck1.EPuck1 method*), 14
set_motor_speeds() (*pipuck.epuck1.EPuck1 method*), 14
set_outer_leds() (*pipuck.epuck1.EPuck1 method*), 14
set_outer_leds_byte() (*pipuck.epuck1.EPuck1 method*), 14
set_right_motor_speed() (*pipuck.epuck1.EPuck1 method*), 14
speaker_enable() (*pipuck.pipuck.PiPuck static method*), 13
switch_to_input() (*pipuck.mcp23017.DigitalInOut method*), 15
switch_to_output() (*pipuck.mcp23017.DigitalInOut method*), 15

T

temperature() (*pipuck.lsm9ds1.LSM9DS1 property*), 18
tof_sensors (*pipuck.pipuck.PiPuck attribute*), 13
ToFSensor (*class in pipuck.tof_sensor*), 15

V

value() (*pipuck.mcp23017.DigitalInOut property*), 15
VL53L1X (*in module VL53L1X*), 18

W

write_data_16() (*pipuck.ft903.FT903 method*), 13
write_data_8() (*pipuck.ft903.FT903 method*), 13

Y

YRLExpansion (*class in pipuck.yrl_expansion*), 14