**Command list for Psi Swarm robot for ROS and serial control.**

By Team Robits

This document lists all functions that can be used in Ros environment or called manually as a serial USB command in order to control the PSI Swarm robot. In order to use it in ROS the provided python file with the functions must be added as a node and calling the functions will give a serial USB message to the robot activating the build in MBed functions for control. Alternatively, the MBed functions can be called manually by sending the trigger command using terminal program like PuTTY. The settings for the Serial USB connections are:

Boud Speed 115200

Data 8 bit with Pause 1bit

**Motors:**

Python function: “forward(variable)”

USB Serial trigger: “forward[decimal number between -1.0 and 1.0]”

Effect: Set the motors to the specified speed (range -1.0 for max. reverse to 1.0 for max. forward)

Returns: “Null”

Python function: “backward(variable)”

USB Serial trigger: “backward[decimal number between -1.0 and 1.0]”

Effect: Sets both motors to the specified inverted speed. Set the motors to the specified speed (range -1.0 for max. forward to 1.0 for max. reverse)

Returns: “Null”

Python function: “brake()”

USB Serial trigger: “brake”

Effect: Enable the active brake on the both motors

Returns: “Null”

Python function:””

USB Serial trigger: “stop”

Effect: Stop both motors. This sets the speed of both motors to 0, it does not enable the active brake.

Returns: “Null”

Python function: “brake\_left\_motor()”

USB Serial trigger: “brakeleft”

Effect: Enable the active brake on the left motor.

Returns: “Null”

Python function: “brake\_right\_motor()”

USB Serial trigger: “brakeright”

Effect: Enable the active brake on the right motor

Returns: “Null”

Python function: “set\_left\_motor\_speed(variable)”

USB Serial trigger: “leftmotor[decimal number between -1.0 and 1.0]”

Effect: Set the left motor to the specified speed. The set motor to the specified (range -1.0 for max. reverse to 1.0 for max. forward)

Returns: “Null”

Python function: “set\_right\_motor\_speed(variable)”

USB Serial trigger: “rightmotor[decimal number between -1.0 and 1.0]”

Effect: Set the right motor to the specified speed. The set motor to the specified (range -1.0 for max. reverse to 1.0 for max. forward)

Returns: “Null”

**Animation:**

Python function:”vibrate()”

USB Serial trigger: “vibrate”

Effect: Make the robot vibrate (turn rapidly left & right) for approximately 1 second with LED flashes. Restores LED states after action

Returns: “Null”

Python function:”red\_run1()”

USB Serial trigger: “ledrun1”

Effect: Patterns LEDs from back to front of robot 3 times then blinks at the front; animation takes about 1 second. Restores LED states after

Returns: “Null”

Python function:”set\_colour(variable)”

USB Serial trigger: “setcolour[whole number between 1 and 3]”

Effect: Sets the colour for single-colour LED animations (default = 1). The colour LED to use in the animation (1 = red, 2 = green, 3 = orange)

Returns: “Null”

**Colour:**

Python function:”start\_colour\_ticker(variable)”

USB Serial trigger: “startcrticker[whole number between 1 and 100000]”

Effect: Starts a polling ticker that cyclically checks to see if a colour can be detected. The approximate cycle period in milliseconds

Returns: “Null”

Python function:”stop\_colour\_ticker()”

USB Serial trigger: “stopcolourticker”

Effect: Stops the polling ticker.

Returns: “Null”

Python function:”detect\_colour\_once()”

USB Serial trigger: “detectcolouronce”

Effect: Function enables colour sensor, takes a reading and identify colour from calibrated colour scores - int range -1 to 8

Returns: “Null”

Python function:”get\_colour\_string()”

USB Serial trigger: “getcolourstring”

Effect: Recognizes and returns a color string from the color score detected. (eg 0="RED")

Returns: “[string of the color]\n”

Python function:”read\_base\_colour\_sensor\_values()”

USB Serial trigger: “readbasecolour”

Effect: Returns the base RGB values of the last detected color.

Returns: “[RGB spectrum]”

**Display:**

Python function:”clear\_display()”

USB Serial trigger: “cleardisplay”

Effect: Clear the display

Returns: “Null“

Python function:”home()”

USB Serial trigger: “home”

Effect: Set cursor to home position

Returns: “Null“

Python function:”write\_string(variable)”

USB Serial trigger: “writestring[message up to 10 char long]”

Effect: Take the 10 character long message after the trigger word and displays the after the cursor position on the display.

Returns: “Null“

**LEDs:**

Python function:”set\_leds(variable1, variable2)”

USB Serial trigger: “setleds[whole number between 1 and 256][whole number between 1 and 256]”

Effect: The numbers represent the 8 LEDs surrounding the robot from its top clockworks. The number is transformed in to its binary format and the 1s indicates that the LED at the current position is turned on. For example, 255 in 8bit is 11111111 which means that all 8 LEDs are on and 3 in 8bit is 00000011 which means that the last 2 LEDs will be turned on.

The first number represent witch LEDs green element will be turned on and the second number shows witch LEDs red element will be turned on. The LEDs with both elements turned on glow in orange.

Returns: “Null“

Python function:”set\_green\_leds(variable)”

USB Serial trigger: “setgreenleds[whole number between 1 and 256]”

Effect: The numbers represent the 8 LEDs surrounding the robot from its top clockworks. The number is transformed in to its binary format and the 1s indicates that the LED at the current position is turned on. For example, 255 in 8bit is 11111111 which means that all 8 LEDs are on and 3 in 8bit is 00000011 which means that the last 2 LEDs will be turned on.

The number represent which LEDs green element will be turned on.

Returns: “Null“

Python function:”set\_red\_leds(variable)”

USB Serial trigger: “setredleds[whole number between 1 and 256]”

Effect: The numbers represent the 8 LEDs surrounding the robot from its top clockworks. The number is transformed in to its binary format and the 1s indicates that the LED at the current position is turned on. For example, 255 in 8bit is 11111111 which means that all 8 LEDs are on and 3 in 8bit is 00000011 which means that the last 2 LEDs will be turned on.

The number represent which LEDs Red element will be turned on.

Returns: “Null“

Python function:”set\_led(variable1, variable2)”

USB Serial trigger: “setled[whole number between 0 and 7][whole number between 1 and 3”]

Effect: Set the state of an invididual outer LED without affecting other LEDs. The first number represents the LED to change state (range 0 to 7) and the second number represents the state of that LED( 0 for off, 1 for red, 2 for green, 3 for orange/both ).

Returns: “Null“

Python function:”set\_base\_led(variable)”

USB Serial trigger: “setbaseled[whole number between 0 and 1]

Effect: Set the state of the base LEDs [if fitted], 0 for off, 1 for on.

Returns: “Null“

Python function:”blink\_leds(variable)”

USB Serial trigger: “blinkleds[whole number between 0 and 1000]

Effect: Turns on all outer LEDs for a period of time defined by number after the trigger word then restore their previous state, that number represents the time (in seconds) to keep LEDs on.

Returns: “Null“

Python function:”set\_center\_led(variable)”

USB Serial trigger: “setcenterled[whole number between 0 and 3]

Effect: Set the state the center LED, 0 for off, 1 for red, 2 for green, 3 for orange

Returns: “Null“

Python function:”set\_center\_led\_brightness(variable)”

USB Serial trigger: “setCledB[decimal number between 0.0 and 1.0]

Effect: Set the brightness of center LED without changing state. The brightness of LED [PWM duty cycle] - range 0.0 to 1.0

Returns: “Null“

Python function:”get\_led\_states()”

USB Serial trigger: “getledstate”

Effect: Returns the current state of the outer LEDs as a 16-bit value when MSB represent the green states and LSB the red states of the 8 LEDs

Returns: “[16-bit value]\n“

**Sensors:**

Python function:”get\_battery\_voltage()”

USB Serial trigger: “getdcvolt”

Effect: Returns the current battery voltage for the robot in Volts. his should range between 3.5V for a discharged battery and 4.2V for a fully charged battery

Returns: “[Float]\n“

Python function:”get\_current()”

USB Serial trigger: “getcurr”

Effect: Returns the current being used by the robot in Amps

Returns: “[Float]\n“

Python function:”get\_temperature()”

USB Serial trigger: “gettemp”

Effect: Returns the temperature sensed by the digital thermometer placed near the front of the MBED socket in degrees ( C ).

Returns: “[Float]\n“

Python function:”get\_dc\_voltage()”

USB Serial trigger: “getbattvolt”

Effect: Returns the voltage sensed from the DC input (post rectification) in voltage (in V). Note some back-voltage from the battery is expected even when no DC input is detected

Returns: “[Float]\n“

Python function:”enable\_ultrasonic\_ticker()”

USB Serial trigger: “enablesonicticker”

Effect: Enables a 10Hz ticker that automatically takes readings from the SRF-02 ultrasonic sensor.

Returns: “Null“

Python function:”disable\_ultrasonic\_ticker()”

USB Serial trigger: “disablesonicticker”

Effect: Disables the ultrasonic ticker (the previas commands).

Returns: “Null“

Python function:”update\_ultrasonic\_measure()”

USB Serial trigger: “updatesonicmeasure”

Effect: Sends a message to SRF-02 ultrasonic sensor (if fitted) to instruct it to take a new reading. The result is available approx. 70ms later.

Returns: “Null“

Python function:”store\_background\_raw\_ir\_values()”

USB Serial trigger: “storebgrawir”

Effect: Stores the raw ADC values for all 8 IR side-facing sensors without enabling IR emitters.

Returns: “Null“

Python function:”store\_illumnated\_raw\_ir\_values()”

USB Serial trigger: “storeillumrawir”

Effect: Stores the raw ADC values for all 8 IR side-facing sensors after enabling IR emitters

Returns: “Null“

Python function:”store\_ir\_values()”

USB Serial trigger: “storeirvalues”

Effect: Stores the raw ADC values for all 8 IR side-facing sensors both before and after enabling IR emitters.

Returns: “Null“

Python function:”get\_background\_raw\_ir\_value(variable)”

USB Serial trigger: “getbgrawir[whole number from 0 to 7]”

Effect: Returns the stored value of the non-illuminated side-facing IR sensor value based on last activation of the sensors. The number shows the index of the sensor to read (range 0 to 7, clockwise around robot from front-right) and returns a unsigned short of background IR reading (range 0 to 4095).

Returns: “[integer]\n“

Python function:”get\_illuminated\_raw\_ir\_value(variable)”

USB Serial trigger: “getillumrawir[whole number from 0 to 7]”

Effect: Returns the stored value of the illuminated side-facing IR sensor value based on last activation of the sensors. The number shows the index of the sensor to read (range 0 to 7, clockwise around robot from front-right) and returns an unsigned short of illuminated IR reading (range 0 to 4095).

Returns: “[integer]\n“

Python function:”calculate\_side\_ir\_value(variable)”

USB Serial trigger: “calculatesideir[whole number from 0 to 7]”

Effect: Returns the subtraction of the background side IR value from the reflection based on last storing of sensors values. For most purposes this is the best method of detected obstacles etc as it mitigates for varying background levels of IR. The number shows the index of the sensor to read (range 0 to 7, clockwise around robot from front-right) and returns unsigned short of compensated ir value (illuminated value - background value) (range 0 to 4095).

Returns: “[integer]\n“

Python function:”read\_illuminated\_raw\_ir\_value(variable)”

USB Serial trigger: “readillumrawir[whole number from 0 to 7]”

Effect: Enables the IR emitter then returns the value of the given side-facing IR sensor. This function is used when just one sensor is needed to be read. The number shows the index of the sensor to read (range 0 to 7, clockwise around robot from front-right) and returns unsigned short of illuminated IR reading (range 0 to 4095).

Returns: “[integer]\n“

Python function:”store\_background\_base\_ir\_values()”

USB Serial trigger: “storebgbaseir”

Effect: Stores the raw ADC values for all 5 base IR sensors without enabling IR emitters.

Returns: “Null“

Python function:”store\_illumnated\_base\_ir\_values()”

USB Serial trigger: “storeillumbaseir”

Effect: Stores the raw ADC values for all 5 base IR sensors after enabling IR emitters.

Returns: “Null“

Python function:”store\_base\_ir\_values()”

USB Serial trigger: “storebaseir”

Effect: Stores the raw ADC values for all 5 base IR sensors both before and after enabling IR emitters.

Returns: “Null“

Python function:”get\_background\_base\_ir\_value(variable)”

USB Serial trigger: “getbgbaseir[whole number from 0 to 4]”

Effect: Returns the stored value of the non-illuminated base IR sensor value based on the last sensor readings. The number shows the index of the sensor to read (range 0 to 4, sensor from left to right viewed from above - 2 is in middle of front) and returns unsigned short of background IR reading (range 0 to 4095).

Returns: “[integer]\n“

Python function:”get\_illuminated\_base\_ir\_value(variable)”

USB Serial trigger: “getillumbaseir[whole number from 0 to 4]”

Effect: Returns the stored value of the illuminated base IR sensor value based on the last sensor readings. The number shows the index of the sensor to read (range 0 to 4, sensor from left to right viewed from above - 2 is in middle of front) and returns unsigned short of illuminated IR reading (range 0 to 4095).

Returns: “[integer]\n“

Python function:”calculate\_base\_ir\_value(int)”

USB Serial trigger: “calculatebaseir[whole number from 0 to 4]”

Effect: Returns the subtraction of the background base IR value from the reflection based on last sensor readings. For most purposes this is the best method of getting uncalibrated values from the base IR sensor as it mitigates for background levels of IR. The number shows the index of the sensor to read (range 0 to 4, sensor from left to right viewed from above - 2 is in middle of front) and returns unsigned short of compensated ir value (illuminated value - background value) (range 0 to 4095).

Returns: “[integer]\n“