Piklet Software Documentation

John Board, 26/07/2015

Contents

Software Overview

The Piklet software is a package that links physical hardware sensors to the Scratch programming environment. It operates in the background of a Raspberry Pi, and connects automatically to Scratch when Scratch is started.

The Piklet software is bi-directional – it can both send and receive messages from Scratch. You send messages by creating an editing variables with specific names. You receive messages from scratch using broadcasts, and sensor blocks from the Sensing panel.

Currently there are 4 sensor drivers written for the Piklet. These include Light/Line, Sound, RFID, and Tilt/G-Force (Accelerometer).

Installing

This section will guide you through installing the Piklet software, and it's dependencies.

Dependencies

The Piklet software requires I2C and SPI are enabled. Follow the following two tutorials to enable them.

Enabling I2C

Enabling SPI

The RPi automatically enables a "serial terminal" – meaning you can talk to the Raspberry Pi using a USB cable. Unfortunately keeping this serial terminal enabled effects sensors connected using the Serial (UART) port. The serial terminal needs to be disabled. Follow the following tutorial to do so:

Disabling UART Terminal

The smbus library for python also needs to be installed, use the following command to do that:

```
sudo apt-get install python3-smbus python3-serial
```

To download the Piklet code, you'll need git and screen – which you can install using:

```
sudo apt-get install git screen
```

Downloading Piklet Software

To download the actual piklket software, type in:

```
git clone https://github.com/Baralabite/Piklet
```

Run as background process

server.py is the "main program file" for the Piklet software. When run, it handles everything. It's designed to run in the background. To make this happen, edit /etc/rc.local, and add the following command:

```
sudo screen -dmLS PikletServer sudo python3 <path_to_server.py>/server.py
```

For example, if I downloaded the Piklet software to /home/pi/Piklet, then the command would be:

```
sudo screen -dmLS PikletServer sudo python3 /home/pi/Piklet/server.py
```

Licensing

The core Piklet software is open source, under the <INSERT LICENSE HERE>. This allows you to <LICENSE ABILITIES>.

Modifying

Because the Piklet software was released as Open Source, you are able to modify and redistribute it as desired. The Piklet was written in Python, and structured so with the intention that you'd be able to add your own sensor code, if desired.

https://github.com/Baralabite/Piklet

All python files (except __init__.py) in the sensors folder are interpreted as sensor drivers.

A template sensor file can be found here:

https://github.com/Baralabite/Piklet/blob/master/sensors/template.py

For more information, continue checking the README file of the repository. Any further queries, please send an email to johnrobboard@gmail.com

Sensor

Before the Piklet software can connect to Scratch, remote sensor connections have to be enabled in scratch. You need to do this each time you start a new scratch program. Follow this tutorial to do that:

Enabling Remote Sensor Connections in Scratch

Before use, sensors need to be plugged in to the Piklet, and then turned on in the Scratch code. To turn on a sensor in the on program, add a variable with a name with the following format.

piklet_<sensorType>_<sensorID>_enabled

Then set the value of the variable to ***1.*** To Turn the sensor off, set the value of the variable to ***0.*** To turn on the RFID sensor, for example, create a variable with the name:

piklet_RFID_uart_enabled

Then set that variable to ***1.***

Overview

Commands

As seen earlier (to turn on the sensor), there are commands that can tell sensors things (for example, turn on or off the sensor). Each command has 3 parts:

piklet_<sensorType>_<sensorID>_<register>

sensorType	This lets piklet's code what type of sensor to be talking to. If you have created your own sensor drivers in the piklet's sensor folder, then you can talk to it by replacing sensorType with the exactly same name as the sensor file. For example, if you have color.py, then you'd access it by going piklet_color_...
sensorID	Sensor ID is used to tell piklet's code what sensor port the specified sensor is plugged in to. For example, GPIO 24, Analog 3, UART, or I2C.
register	Each sensor driver as a number of settings that can be changed, for example whether it's enabled or not. This setting is called a register, and by editing the variable, you change the value of that specific register.

Here's a list of valid inputs for sensorType and sensorID. Consult the specific sensor's documentation to find a list of registers:

sensorType	**sensorID**	**Description**
RFID	uart	RFID sensor, plugged into the UART plug
light	<Any valid GPIO pin, that the light sensor is plugged in to. For example "24">	Light sensor. Example turning light sensor on could be piklet_light_24_enabled
accelerometer	i2c	Accelerometer. Measures G-Force.
sound	<Any valid analog pin. From 0 to 3>	Sound sensor. piklet_light_3_enabled

RFID

The RFID sensor can read RFID cards/tags, such as student ID cards, go cards, etc.

Physical Setup

5V	Plugged into RPi 5v
GND	Plugged into RPi GND
TX	Plugged into RPi Serial RX (UART)
RX	Plugged into RPi Serial TX (UART)

Registers

Register	**Inputs**	**Description**
enabled	0/1	Turns the sensor on/off

Example Usage

1. Create variable to piklet_RFID_uart_enabled
2. Set variable to 1
3. Wait until you receive a broadcast “rfid-updated”
4. Read values from sensor rfid

Known Bugs

Each scan is 1 card behind. For example, if I scan card 1, it won’t read. If I scan card 2, it’ll read as card 1. If I scan card 1 again, it’ll read card 2, etc.

Troubleshooting

Light (QTI Sensor)

The RFID sensor can read RFID cards/tags, such as student ID cards, go cards, etc.

Physical Setup

R(ed)	Plugged into RPi 5v
B(lack)	Plugged into RPi GND
W(hite)	Plug into RPi GPIO

Registers

Register	**Inputs**	**Description**
enabled	0/1	Turns the sensor on/off
threshold	Integer	Amount of change required before an updated value is sent to Scratch – for example if the number keeps moving between 88 and 89, you don't want it to keep messaging Scratch. Set the threshold 2 so the value would have to move from 88 to 90 before it'd alert Scratch

Example Usage

1. Create variable to piklet_light_<gpioPin>_enabled
2. Set variable to 1
3. Wait until you receive a broadcast "light-updated"
4. Read values from sensor light

Known Bugs

Sound

The RFID sensor can read RFID cards/tags, such as student ID cards, go cards, etc.

Physical Setup

AO	Plug into RPi Analog Input
G	Plug into RPi GND
+	Plug into RPi 5V
DO (Doesn't need to be connected)	Plug into RPi GPIO

Registers

Register	**Inputs**	**Description**
enabled	0/1	Turns the sensor on/off
threshold	Integer	Amount of change required before an updated value is sent to Scratch – for example if the number keeps moving between 88 and 89, you don't want it to keep messaging Scratch. Set the threshold 2 so the value would have to move from 88 to 90 before it'd alert Scratch

Example Usage

1. Create variable to piklet_sound_<analogPin>_enabled
2. Set variable to 1
3. Wait until you receive a broadcast "sound-updated"
4. Read values from sensor sound

Known Bugs

You have to be very close to the sensor for it to work.

Accelerometer

The RFID sensor can read RFID cards/tags, such as student ID cards, go cards, etc.

Physical Setup

VCC/5V	Plug into RPi 5V
SCL	Plug into RPi SCL (I2C)
SDA	Plug into RPi SDA (I2C)
GND	Plug into RPi GND

Registers

Register	**Inputs**	**Description**
enabled	0/1	Turns the sensor on/off
threshold	Integer	Amount of change required before an updated value is sent to Scratch – for example if the X axis keeps moving between 88 and 89, you don't want it to keep messaging Scratch. Set the threshold 2 so the value would have to move from 88 to 90 before it'd alert Scratch

Example Usage

1. Create variable to piklet_accelerometer_i2c_enabled
2. Set variable to 1
3. Wait until you receive a broadcast "accelerometer-updated"
4. Read values from accelerometer-x/y/z sensor