AGILE Maker's Shield

User manual
V0.2

AGILE - Adaptive Gateways for dIverse muLtiple Environments
H2020 Project - 688088
https://agile-iot.eu/

European Commission
Horizon 2020
European Union funding
for Research & Innovation

Abstract

This document describes the main features of the AGILE Maker's Shield and the necessary steps to use them.

Advanced information can be found in the corresponding repositories on Github for AGILE Maker's Shield:

Hardware https://github.com/Agile-IoT/agile-makers-shield-hardware

Firmware https://github.com/Agile-IoT/agile-makers-shield-firmware

Software https://github.com/Agile-IoT/agile-makers-shield-software

Document History

Version	Date	Comments
V0.1	10/04/2017	Document creation
V0.2	11/05/2017	Firmware and software sections added
V0.3	29/11/2018	DBus specification

License

Contents

List of Figures

List of Tables

1 Introduction

The AGILE platform is an integrated solution for IoT application that exploits the benefits provided by a modular hardware platform and a pervasive software framework that complements each other.

The AGILE Platform is based on two different versions of a modular gateway:

- The makers-friendly version, for easily and fast prototyping of IoT solutions.
- The industrial version, aimed to M2M, IoT and industrial markets.

This document provides a deep description of the AGILE Maker's Shield, while the industrial version is just named for reference and it will be described out of this document.

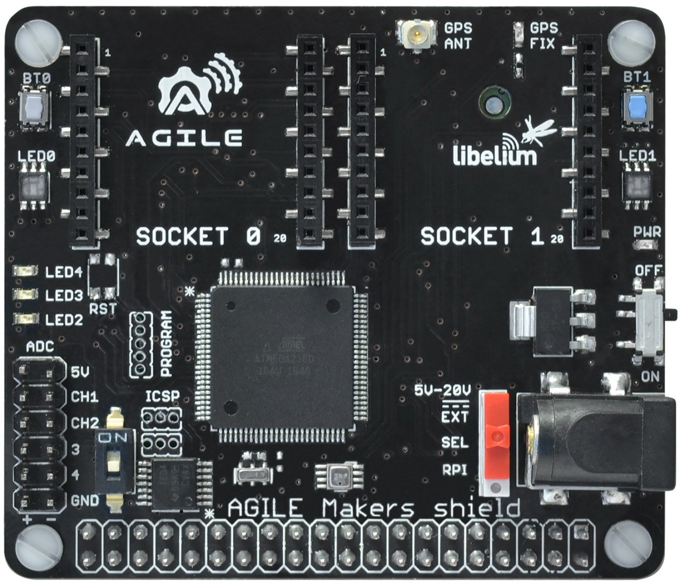

Figure 1: AGILE Maker's Shield

2 Hardware

The AGILE Maker's Gateway is composed of a Raspberry Pi, a shield (also called hat) and different communication modules plugged both to the HAT and directly to the RPi.

The main structure of the maker's shield is thought to provide two expansion radio sockets for the RPi platform, with the necessary power and communication capabilities for the most popular IoT radios on the maker's community.

The expansion radio sockets are managed by a microcontroller, which takes control of each one separately and can turn on or off each one when needed. Besides that, the microcontroller also manages the other shield features like LEDs, buttons etc.

This shield to be connected on top of the RPi is compliant with the HAT [1] ("Hardware Attached on Top") specification for RPi [2], which defines the standard to be followed by shields designed for RPi.

Many shields (or add-on boards) to be plugged on top of RPi can be found at the market. However, not all these shields comply with the HAT standard. Since our add-on board has been designed according to it, in this documentation we will refer indistinctly to it as shield or hat.

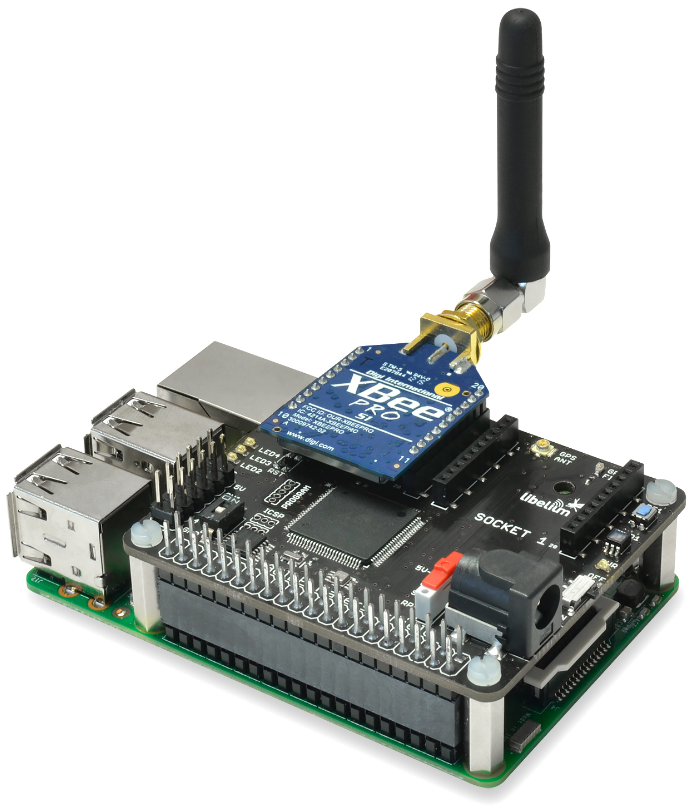

Figure 2: AGILE Maker's Gateway

[1] https://www.raspberrypi.org/blog/introducing-raspberry-pi-hats/
[2] https://github.com/raspberrypi/hats

2.1 Dimensions

The PCB dimensions of this shield meet the mechanical requirements defined in the HAT specification. The display and camera slots are not implemented as they are optional requirements. The figure 3 shows the physical dimensions of the maker's shield in millimetres.

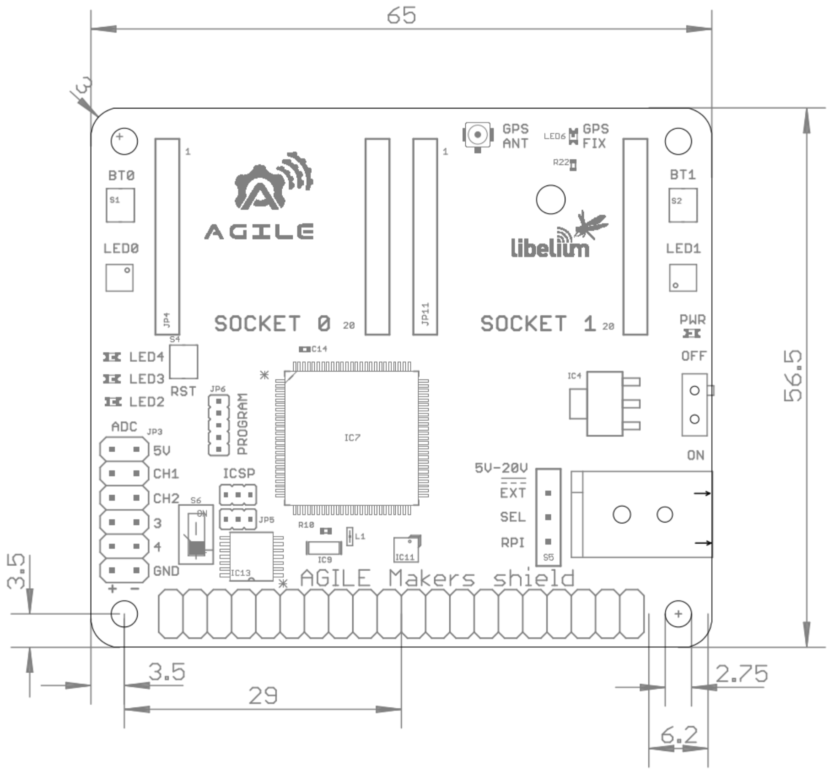

Figure 3: HAT dimensions

There are four mounting holes on each corner to allow securing the shield to the RPi using the provided hexagonal spacers and M2.5 screws, as shown in figure 4.

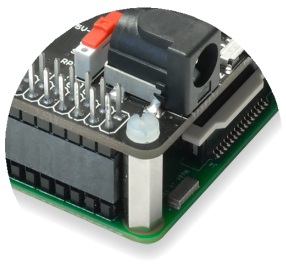

Figure 4: Detail of spacers to secure the shield to RPi platform.

2.2 Architecture

The figure 5 shows the main parts of the shield. Each part is fully described in the following subsections.

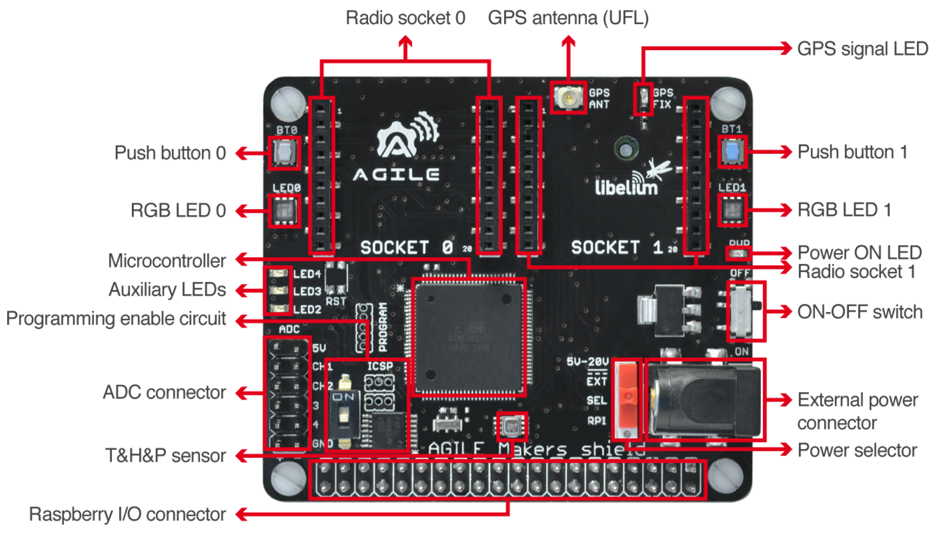

Figure 5: Maker's shield main blocks

2.2.1 Microcontroller

The microcontroller is the brain of the shield and its main task is to free the RPi from managing the rest of the shield modules, like radios, sensors or LEDs.

The microcontroller has a dedicated UART bus for serial communication with each radio socket and the GPS module. Besides it is connected to RPi through the I^2C bus, which is also used by the ADC and the on-board sensors.

Only one RPi GPIO pin (GPIO4) is used by the shield, so the rest are free for custom applications.

2.2.2 Communication sockets

The shield allows connecting two radio modules at the same time by using the two XBee form factor sockets (Radio sockets 0 and 1 on the figure 5). The XBee form factor has been chosen because its popularity between the maker's community and the big amount of IoT radios already designed in this way.

Each socket has a 3V3 supply which can be switched on or off independently, allowing save power turning off the radios when they are not used.

Besides, the UART communication bus is available on both sockets, making easy to integrate most of the IoT radio modules on the market.

Finally, each socket has a push button and a dedicated RGB LED, to allow "hotswapping" radio modules while the RPi remains on and also auto-detection of the new radio. For example, if the user wants to replace the radio on socket 0, it is only needed to unplug the current radio module, plug the new one and press the button. The new radio will be detected by AGILE, without turning off the RPi.

2.2.3 Power stage

The maker's shield can be powered in two ways:

- From the 5V supply pin of the RPi.
- Powered by an external DC source through a jack connector (from 5V to 20V).

There is a power selector to choose between the two powering options. By default, most of the radios will work with the 5V supply of the Raspberry without any problem. The external power source option is left open just in case a radio module could need more power than that available on the 5V pin.

Either ways of powering are protected against over currents with a resettable fuse. After the fuse, there is a slide switch (see figure 6) to turn on and off the whole shield. The on position is the nearest to the external power socket.

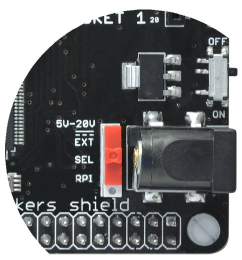

Figure 6: Maker's shield power stage

2.2.4 LEDs

The maker's shield has some LEDs to allow the user to know the current state just by visual inspection. Besides, each LED is programmable and also has dimming capability, except for the on-off and the GPS LEDs.

On-off LED It is connected directly to the 3V3 supply, so it will light if the shield is plugged into a RPi and the slide switch is in on position.

Socket LEDs Each radio socket has its own LED to indicate different status. These LEDs are RGB.

Auxiliary LEDs There are three auxiliary LEDs that can be programmed by the user freely.

GPS LED There is a dedicated LED to indicate if the GPS position is fixed. This LED is directly managed by the GPS module, so it can't be programmed or dimmed.

2.2.5 ADC

The maker's shield includes a 18-Bit, Multi-Channel $\Delta\Sigma$ Analog-to-Digital Converter to allow the integration of external analog sensors with high accuracy. It is powered at 5V and tied to GND, so do not connect voltages above this range directly to the ADC pins or the ADC can be damaged.

A dedicated connector is provided to allow accessing the four differential ADC channels, besides than two power pins and two ground pins. The on-board 2.048V reference voltage enables an input range of $\pm$2.048V differentially (the full scale range is 4.096V/PGA).

2.2.6 On board Temperature, Humidity and Pressure sensor

The maker'shield includes an on board digital temperature, humidity and pressure sensor developed by Bosh.

2.2.7 GPS

The shield includes an on board GPS chip connected directly to the microcontroller. Despite its placement on the bottom layer of the shield due to mechanical restrictions, an UFL connector is included to allow connecting an external antenna.

Besides, there is a LED that will blink while the GPS is getting position and it will turn off when the position is fixed.

2.2.8 Programming circuitry

To easily update the shield firmware from the RPi, some additional circuitry is included. This avoids the need of external tools to update the firmware, making the shield more flexible and allowing the user to upload custom firmwares.

The firmware update can be enabled by setting the DIP switch to the ON position. After the update, the circuitry can be disabled by switching back the DIP switch.

2.2.9 EEPROM

An EEPROM memory has been added according to the HAT specification. Besides, ID_SC and ID_SD pins have been routed only to the EEPROM memory, and pulled up to 3K9 resistors. No other devices are attached to these pins, as it is shown in figure 7.

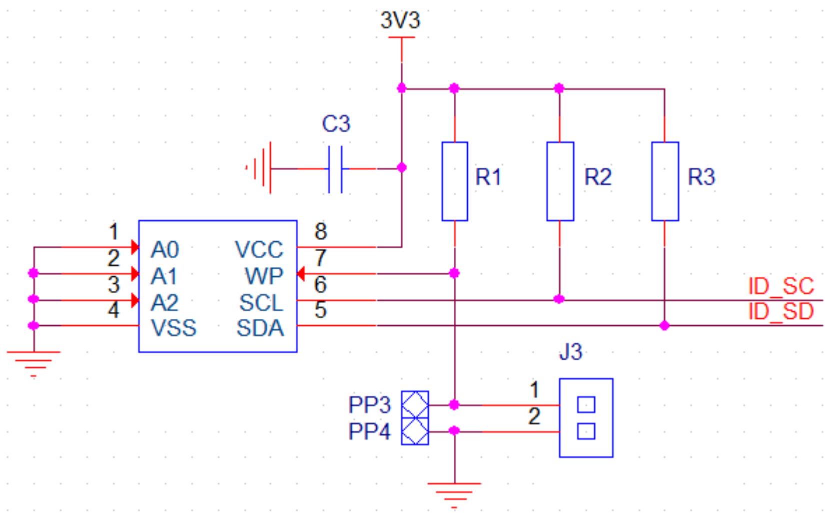

Figure 7: EEPROM circuit following the RPi HAT specification

The ID EEPROM file will be generated using the official tool from Raspberry and the contents will follow its format. The values used are show in the table 1

Vendor Info		
product_uuid	00000000-0000-0000-0000-000000000000	
product_id	0x0001	
product_ver	0x0001	
vendor	"Libelium Comunicaciones Distribuidas S.L."	
product	"AGILE Gateway Hat - Maker's version"	
GPIO Settings		
gpio_drive	0	
gpio_slew	0	
gpio_hysteresis	0	
back_power	0	
GPIO Functions		
GPIO4	INPUT	DEFAULT

Table 1: EEPROM values

3 Firmware

The firmware for the AGILE Maker's Shield consist on a program that tells the microcontroller how to communicate with the rest of the parts of the shield.

The main task of this board is to expose the I^2C bus of the microcontroller to the Raspberry Pi I^2C bus. Through this bus, the RPi can share data with the microcontroller.

The microcontroller will use its own UART ports to check for input events on the sockets and will store the data in its own memory waiting for the RPi to read it. It will also redirect the data sent from the RPi to the sockets. Also, one of the UARTs is connected to the GPS, so the microcontroller will store the last data from it and update the GPS data if ordered from the Rpi. At last but not least, the microcontroller will also control the LEDs and the buttons.

If the microcontroller must notify the RPi because of an event (for instance there is new data from one of the UARTs or a button was pressed) it will put on high state the pin attached to the RPi GPIO4, so a handler for interruptions can be used on the RPi.

The sources for the firmware, along compiled binaries and scripts to load them can be found in the AGILE Maker's Shield Firmware repository.

3.1 Installation

In order to install the firmware, follow the steps of the repository's README. A summarized version of these steps is:

- Install the *avrdude* program in the Raspberry Pi
- Use the default configuration for *avrdude*.
- Copy the *load_agile_firmware.sh* script and the firmware binary.
- Put the shield programming switch in the ON position.
- Execute the script to load the firmware.
- Put the shield programming switch in the OFF position. The LEDs will blink if all went OK.

3.2 Build from sources

Again, in order to build the firmware from the sources, follow the steps of the repository's README.

Along the source code there are some files to define a new board in the Arduino IDE, so the firmware can be compiled and the binary can be generated using it.

3.3 ID EEPROM

The repository also contains the files needed to generate a valid HAT ID following the Raspberry Pi HAT ID EEPROM Format Specification.

3.4 I^{2}C Addresses and regiters

The I^{2}C bus shares the addresses of three devices. These addresses can be found in the table 2.

Device	Address
Microcontroller	0x14
MCP3424	0x6C
BME280	0x77

Table 2: I^{2}C addresses

The registers for MCP3424 and BME280 are out of scope of this document, as they are manufactured by other vendors and all their technical data and documentation can be found in their respective datasheets [3] [4].

Regarding the microcontroller, a custom set of registers has been designed. These registers and their functions are described in the table 3.

	Device	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	Socket 0	FIFO Tx	FIFO RX	FIFO Available		FIFO To read		Baudrate				Databits	Stopbits	Parity	Status	ISR UART	ISR Button
10	Socket 1	FIFO Tx	FIFO RX	FIFO Available		FIFO To read		Baudrate				Databits	Stopbits	Parity	Status	ISR UART	ISR Button
20	GPS	Update	Buffer size	Last GGA	Last RMC												
30	LEDs	S0 R	S0 G	S0 B	S1 R	S1 G	S1 B	Aux 2	Aux 3	Aux 4							
40																	
50																	
60																	
70																	
80																	
90																	
A0																	
B0																	
C0																	
D0																	
E0																	
F0																	Check byte

Table 3: Microcontroller I^{2}C registers

4 Software

The purpose of the AGILE Maker's Shield software is to expose all the functions and features of the shield over D-Bus [5]. The software is split in two parts: one part communicates and controls the diverse features of the shield, like the GPS, the ADC or the LEDs, and the other part manages the different communication modules that can

[3]http://www.microchip.com/wwwproducts/en/MCP3424
[4]https://www.bosch-sensortec.com/bst/products/all_products/bme280
[5]https://www.freedesktop.org/wiki/Software/dbus/

be used with the shield. Both parts are controlled by the D-Bus server, which translates the D-Bus paths and objects to their corresponding sections of code in the libraries. Everything is programmed with Python3.

In the following subsections a brief explanation of the different parts and inner workings of the software can be found. The code referenced can be found in the AGILE Maker's Shield Software repository. For instructions on installation, execution and examples of use, please refer to the README of the repository.

4.1 D-Bus

The different functions are exposed over D-Bus in paths and objects:

- iot.agile.Protocol
  - /iot/agile/Protocol/XBee_802_15_4/socket0
  - /iot/agile/Protocol/XBee_802_15_4/socket1
  - /iot/agile/Protocol/XBee_ZigBee/socket0
  - /iot/agile/Protocol/XBee_ZigBee/socket1
  - /iot/agile/Protocol/LoRaWAN/socket0
  - /iot/agile/Protocol/LoRaWAN/socket1
- iot.agile.Feature
  - /iot/agile/Feature/ADC
  - /iot/agile/Feature/Atmospheric_Sensor
  - /iot/agile/Feature/GPS
  - /iot/agile/Feature/LEDs

After detecting one of the possible modules in SOCKET_0 or SOCKET_1, a Dbus signal is sent over Dbus to the corresponding interface name and object path. The signal include the configuration parameters in its arguments.

The interface name in all the cases is BUS_NAME = iot.agile.Protocol . The object path changes depending on the module and the socket where it has been detected.

Regarding the arguments included in the sending of the signal are the following in this order: BAUDRATE, DATABITS, STOPBITS and PARITY. The values each one of them can take and the correspondence with the configuration parameters are the following:

	Value received in the signal	**Value of the configuration parameter**
Baudrate	600	600
	1200	1200
	2400	2400
	4800	4800
	9600	9600
	19200	19200
	38400	38400
	57600	57600
	115200	115200
Databits	0x00	5
	0x02	6
	0x04	7
	0x06	8
Stopbits	0x00	1
	0x08	2
Parity	0x00	None
	0x20	Even
	0x30	Odd

4.2 I^{2}C bus

There are some libraries to control the communications over the I^{2}C bus, one for each device of the bus.

In the case of the microcontroller library, it will always return the same instance when calling the class constructor, and every method which make calls to the device has a lock mechanism, so even when instantiating the class and calling methods from multiple threads, one call to the device won't start until the previous one has finished all its operations.

4.3 Serial bus

The serial class emulates the original pySerial [6] library for Python, so the code still works if the library is changed to pySerial and the modules are connected to a serial port. Nevertheless, if using the AGILE Maker's Shield, this library will handle the communications with the different protocols. It will also handle the interruptions from the different events of the shield (new data in any socket or any socket button pressed).

To avoid missing some events, the class that manages the interruptions will return always the same instance when calling its creation method. Other classes can subscribe to it in order to receive notifications when there is a new ISR.

[6]https://pypi.python.org/pypi/pyserial

5 Getting started

To start using the shield you will need, at least, these items:

- A Raspberry Pi 2 (or newer)
- An AGILE Maker's Shield
- A micro SD card with the AGILE image
- Power adapter for Raspberry (5V, 2A)
- An Ethernet cable to provide internet to the Raspberry
- If SSH access is not used then a keyboard, a mouse, an HDMI cable and a monitor are needed

Optional items:

- External DC power source (from 5V to 20V) with male jack of 2mm
- Radio modules
- GPS antenna with UFL connector, if your application requires GPS
- Spacers and screws to secure the shield to the RPi
- A case for the Raspberry Pi

Note: The AGILE Maker's Shield v007 has the external power feature disabled. Besides, to select the RPi power option, the small white hole near the RPi label must be visible (Red selector up), just as it is shown in the figure 8.

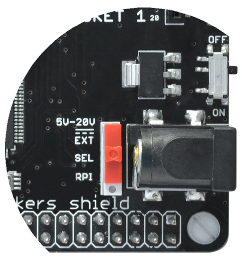

Figure 8: Maker's shield powered from the RPi

Before turning on the RPi, plug the shield carefully on the Rpi. Then, if the application requires that, place the spacers and the screws on the mounting holes to secure the shield to the RPi.

Then check that the shield has the switches in the desired positions:

- On-off switch should be off.
- External power selector should be in “Rpi” position.

After these checks, power on the RPi and once the AGILE environment is loaded, turn on the shield.

If you are going to update the firmware, remember to select the “ON” position on the programming DIP switch.

If the AGILE shield needs to be detached from the RPi platform, take special care on the RPi pins. It can be bent easily if too much force is applied to detach the shield.