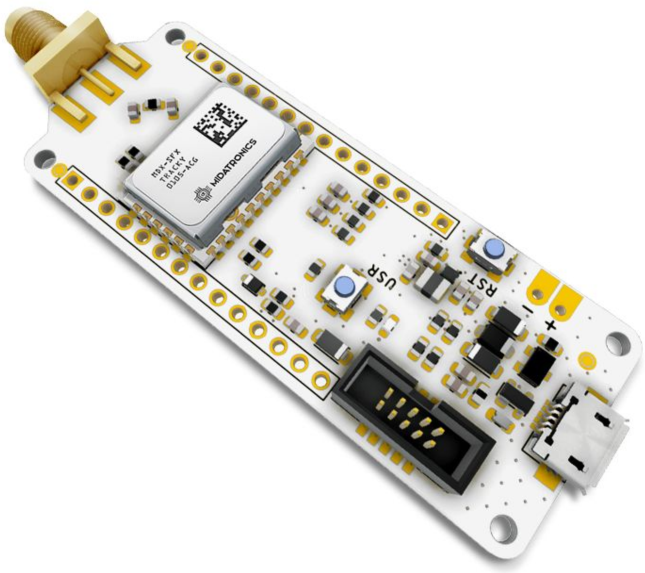

TRACKY MKR

User’s Guide

MDX-MKR-SFXO-01 : Tracky MKR

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Midatronics S.r.l. without notice.

Outline

Illustrations

Tables

Revisions

REVISION	DATE	DESCRIPTION	STATUS	AUTHOR	REVISER
Ver 1.0	03/16/2019	First Release	Draft	info@midatronics.com	UA-EM
Ver 1.1	06/27/2019	Updated Power info	Draft	info@midatronics.com	

Disclaimer

Midatronics S.r.l.
Via Zucchi 1 20900
Monza (Monza Brianza)
Italy

info@midatronics.com
www.midatronics.com

1 Introduction

1.1 Description

This document describes the Tracky MKR Board.

Tracky MKR board is based on the Tracky Open module that contains an STMicroelectronics STM32L072KZ, a single-core MCUs based on an Arm® Cortex®-M0+ core running at 32 MHz.

The board pinout is compatible with Arduino MKR boards, and can be programmed with Arduino IDE thanks to the STM32Duino project. The processor voltage is 3.3V .

Onboard SWD connector allows programming the board with STLink in-circuit debugger and programmer and Atollic/IAR/SW4STM32/Keil IDEs.

Main features

- Board size 65.90 x 25 mm
- Processor Voltage: 3.3V

1.2 Kit Contents

The following items are included in the box:

-

1.3 Getting Started

The Tracky MKR board, developed by Midatronics for Arrow Electronics, is a ready-to-use Internet of Things (IoT) hardware.

Please refer to software chapter to learn how to get started to develop your application using the Arduino IDE or STM32 Studio IDE.

2 System Overview

2.1 Sigfox LPWAN

From : https://ubidots.com/blog/explaining-sigfox/

Sigfox is a Low Power Wide Area Network (LPWAN) technology specially designed for the Internet of Things. Devices connected using SigFox consume **little power** and operate over **large distances** compared to WiFi and Bluetooth connection protocols which consume *more* power and work best in short range. The chronology of a SigFox application follows these three basic steps:

1. Numerous objects (devices) connected to the Internet send data through the SigFox network to a SigFox base station (gateway).
2. The base station then detects, demodulates, and reports the messages to the SigFox cloud across 3 channels, at least every 10 minutes.
3. The SigFox cloud then pushes these messages to many customer servers and IoT platforms based on the client's application.

Technically, the SigFox network differs from other LPWAN networks in the methods it sends data and the electrical guidelines that govern the quantity, speed, and duration of the data being sent. SigFox is most used for **low-power** applications that only require sending **small amounts of data, infrequently, over large distances**. Perfect for Agro environments and asset management across vast distances.

2.1.1 How Sigfox works?

The SigFox network consists of these *elements*:

- Objects (devices)
- Base stations (gateways)
- Cloud (internet)

The diagram below illustrates the key *elements* of data transmission via SigFox:

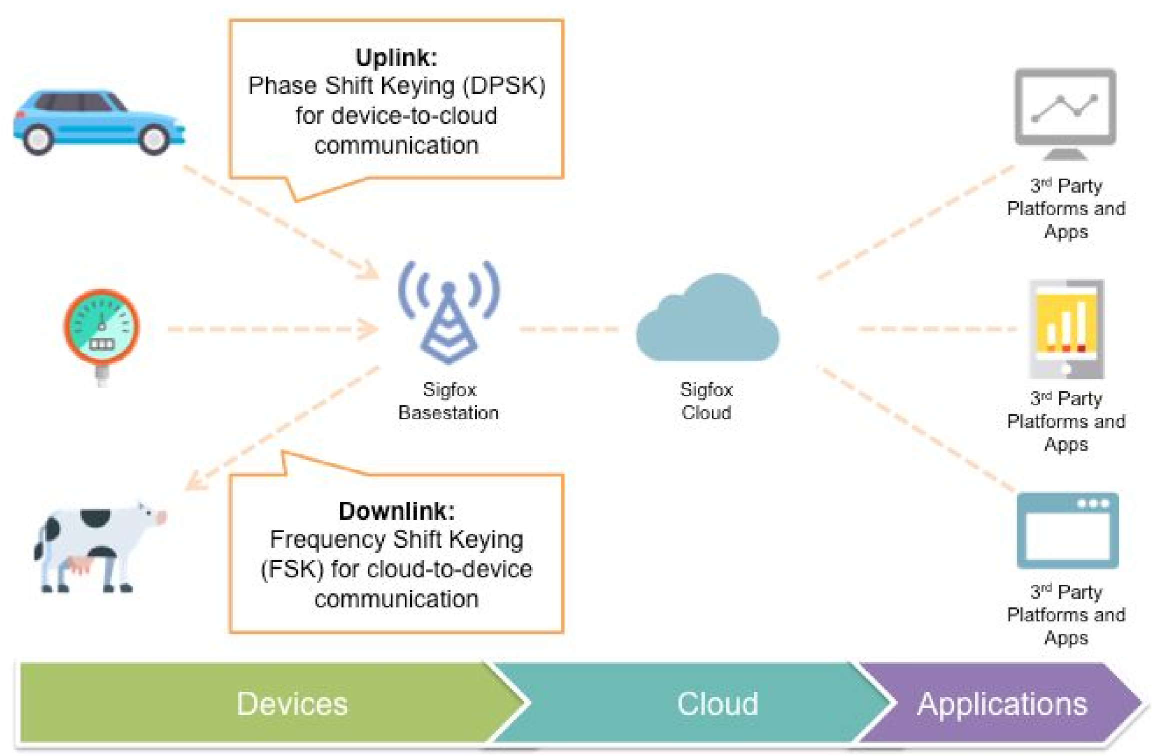

Figure 1. Sigfox Key elements

DPSK is a method used by base stations (gateways) to convert a signal, debug it, and forward to the cloud for processing.

SigFox data transmission can be better understood as follows: Objects (devices) are connected to the internet using the SigFox network. The object can be a temperature, humidly, and/or saturation (etc.) sensors located within 1,000 meters of a base station (gateway).Sigfox uses **Phase Shift Keying (DPSK)** for device-to-cloud communication, or "uplink", and **Frequency Shift Keying (FSK)** for cloud-to-device communication, or "downlink".

2.1.2 What is DPSK?

DPSK is a method used by base stations (gateways) to *convert* a signal, *debug* it, convert it back to be sent to the cloud. When a signal travels from a device to a base station, it inevitably encounters interference from the environment (think rain or dense forests). Interference is universal; any signal from any internet network will become impaired and look slightly different upon reaching its destination. SigFox alleviates this problem by utilizing DPSK. The role of DPSK is to make sure that the signal that leaves the base station is the same exact signal that left the device. The base station hardware accomplishes this by shifting the signal’s phase to discover and eliminate impairments. SigFox hardware at the base stations accomplishes this by:

1. Object sends data to the base station in the form of digital bits. A “high” pulse occurs when there’s a 1, and a “low” pulse occurs when there’s a 0. Here is an input digital bit stream 1 1 0 0 0 1 1 0:

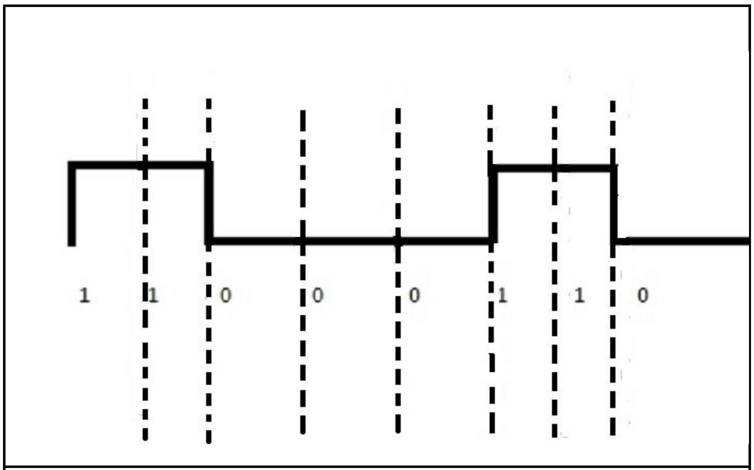

Figure 2. Digital bit stream

2. This bit stream is then converted into a *different* sequence of 1’s and 0’s as it passes through the demodulator circuit. The new sequence is not arbitrary, rather carefully calculated using sophisticated hardware. The purpose of this conversion is to prepare the signal for electrical analysis. Whenever the state of the input signal goes from high to low (1 to 0), the hardware shifts the phase of the signal. To phase shift, a signal simply means to impose a time gap between the signal’s original and new paths. Once the phase is shifted, a signal will either lag/lead where the original path once was:

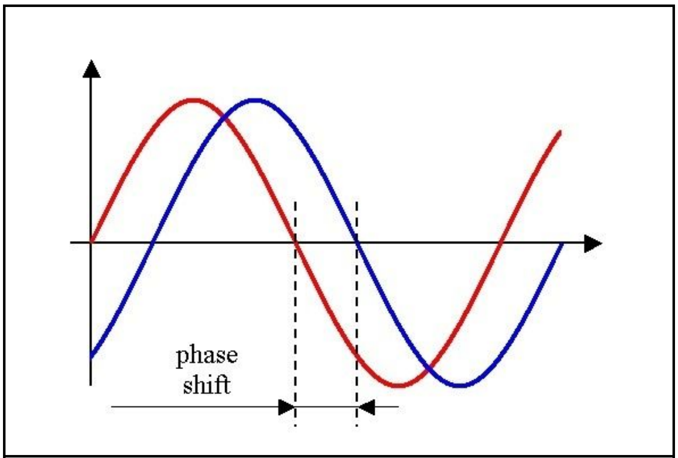

Figure 3. Phase Shift Keying

3. The more a signal at the base station becomes phase shifted, the more exposed its impairments become. Analogously, the more frequent an injured person visits the hospital, the more X-rays the doctors take to better understand and correct the injury. When a signal is “injured” from the environment, the injury is not noticeable until the signal is phase-shifted and run through the “X-ray” circuitry which analyzes these phase shifts, discovers where the impairments exist, and subsequently “cleans” the data for transmission. In summary, the base station hardware shifts the phase in order to take an “X-ray” of the data to diagnose what interference/impairments exist and how it is to be corrected.
4. Following the phase, the hardware circuitry converts the original signal back to its basis sequence, but without the impairments.

When the cloud receives an uplink signal from the base station, it will respond with a downlink signal to the device. Downlink signals use Frequency Shift Keying.

2.1.3 What is Frequency Shift Keying?

Frequency Shift Keying (FSK) is similar to Differential Phase Shift Keying (DPSK) in the sense that both processes convert the input signal, analyze/discover impairments, eliminate them, and convert the data back to the original signal. However, instead of shifting and analyzing the phase, FSK shifts and analyzes the frequency. Just like phase shifts in DPSK, the frequency shifts in FSK expose the signal's impairments where sophisticated circuitry can debug them. The outstanding question now becomes, why does SigFox use DPSK for uplink transmission and FSK for downlink?

1. DPSK is more bandwidth efficient than FSK so it has fewer frequencies and channels available to transmit the signal.
2. Less “space” to transmit the signal = lower data rate and throughput
3. Lower data rate = more sensitive receiver (like a base station) of the signal
4. Higher sensitivity = more achievable range. ie. data from sensor devices can be detected from farther away.
5. Uplink signals typically encounter more interference than downlink signals, so having a narrow bandwidth in DPSK = power is more concentrated = more robustness to interference
6. Since interference is not as big as a concern for downlink, downlink signals are more focused on reaching as many applications as efficiently as possible. In FSK, more bandwidth = more space to send a signal = more reachable applications

2.1.4 Conclusion

The technology used by SIGFOX contributes a long-range, low-power, low throughput communications network with excellent protection from environmental interference allowing data to reach many applications effectively. SIGFOX is still in the “early adopter stage” for connectivity solutions; however, there are already many millions of connected devices around the world with Sigfox technology proving it has the potential to provide a cost-effective solution in a variety of markets and industries.

2.2 S2-LP ULP Transceiver

STMicroelectronics S2-LP Ultra-Low Power, High Performance, Sub-1GHz Transceivers are designed for RF wireless applications. The S2-LP Transceivers operate in the license-free ISM and SRD frequency bands at 433MHz, 512MHz, 868MHz, and 920MHz. The Transceivers can also be programmed to operate at other additional frequencies in the 413MHz to 479MHz, 452MHz to 527MHz, 826MHz to 958MHz, and 904MHz to 1055 MHz bands.

The S2-LP supports the 2(G)FSK, 4(G)FSK, OOK, and ASK modulation schemes. The air data rate is programmable from 0.1kbps to 500kbps. The S2-LP Transceiver can be used in systems with channel spacing down to 1kHz enabling the narrow band operations.

The S2-LP shows an RF link budget higher than 140dB for long communication ranges and meets the regulatory requirements applicable in territories worldwide, including Europe, Japan, China, and the USA.

For more informations on S2-LP visit the following site:

https://www.st.com/b/en/wireless-transceivers-mcus-and-modules/s2-lp.html

2.3 Block Diagram

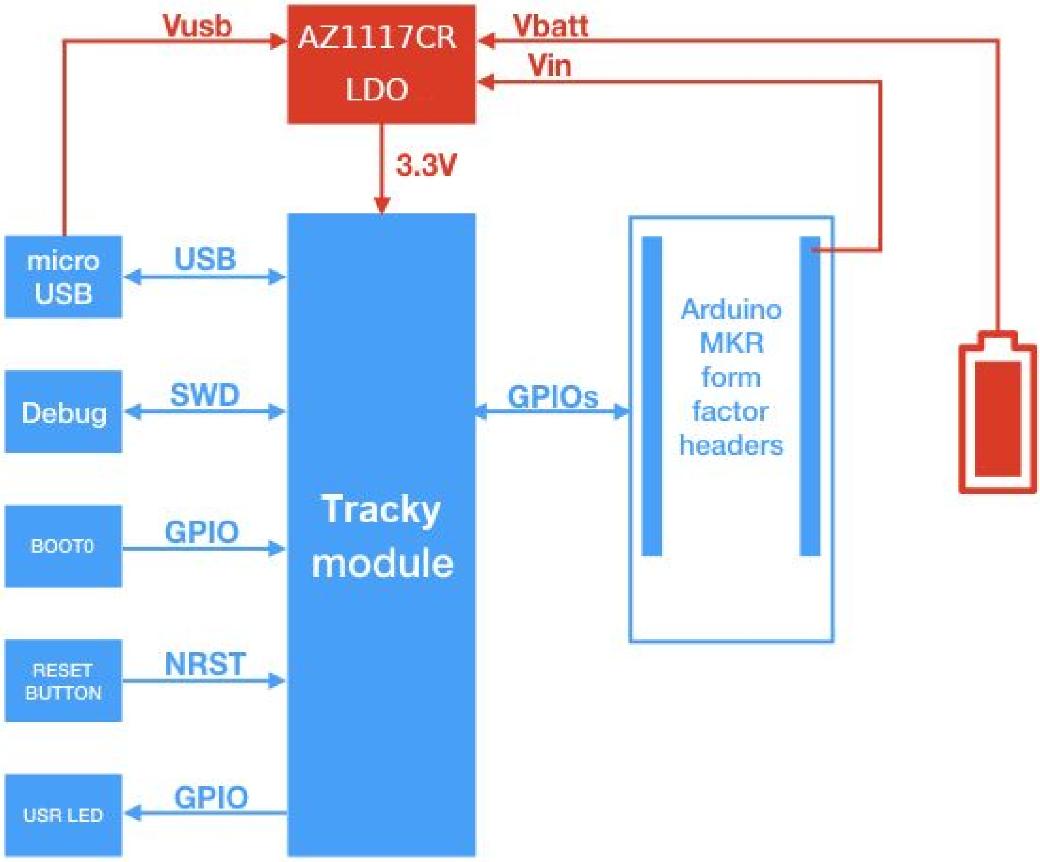

Figure 4. Tracky MKR Block Diagram

2.4 Board Specifications

Characteristics	Value
CPU Clock Speed	32 MHz
Flash Memory	192 Kbyte
SRAM	20 KByte
EEPROM	6 KByte
Connector	1 USB 1 SWD Debugger 1 battery Arduino MKR compatible pinout
Board supply voltage	3.3 V to 5.5 V DC
Operating Voltage	3.3 V (*)
Operating Temperature	-40 °C to +85 °C
Dimensions	65.90 x 25 mm
RoHS status	Compliant

Table 1. Board Specifications

(*) All digital I/O refer to this level

2.5 Tracky Open Module Block Diagram

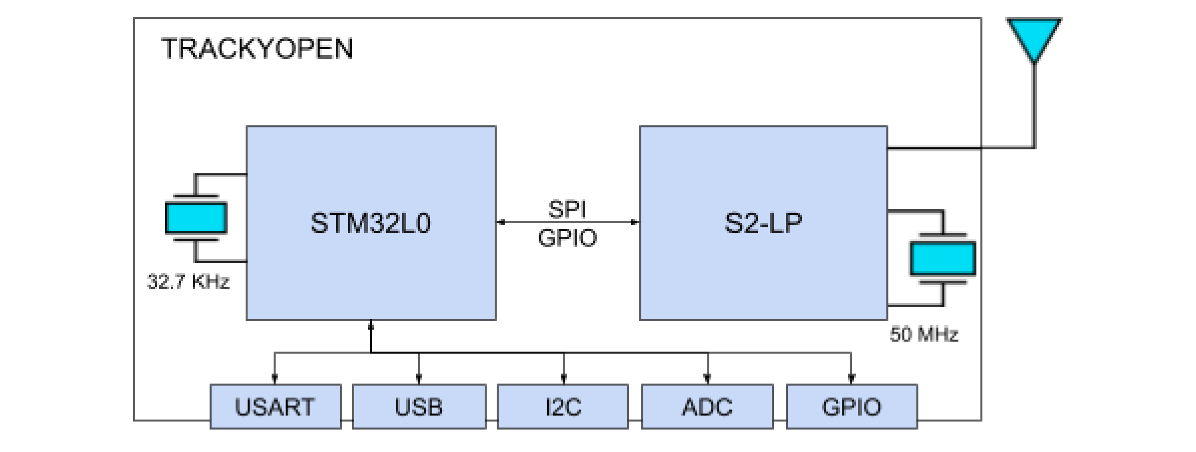

Figure 5. Tracky Open Module with PCB Antenna / or uFL connector

2.6 Tracky Open MCU Configuration

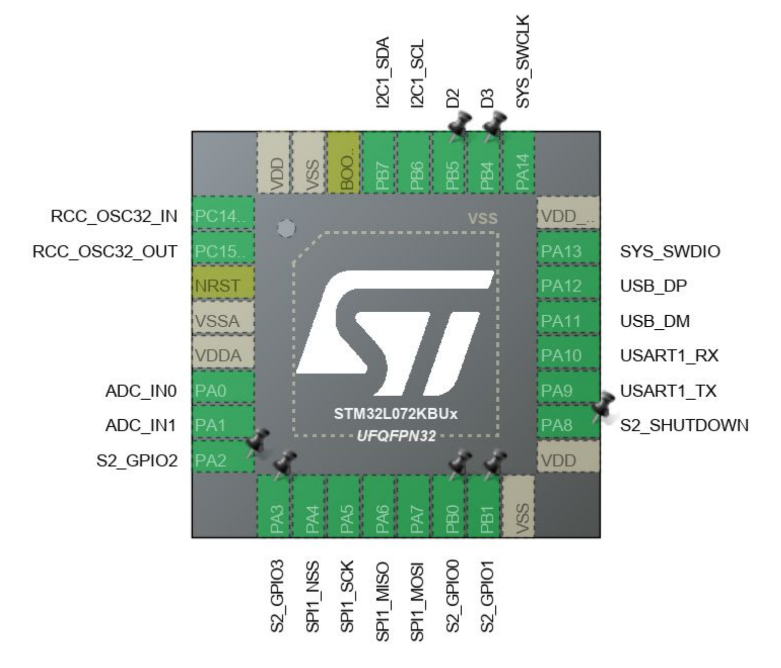

Figure 6. CubeMX MCU Configuration

3 Connectors

This chapter gives you an overview of the Tracky MKR board connectivity.

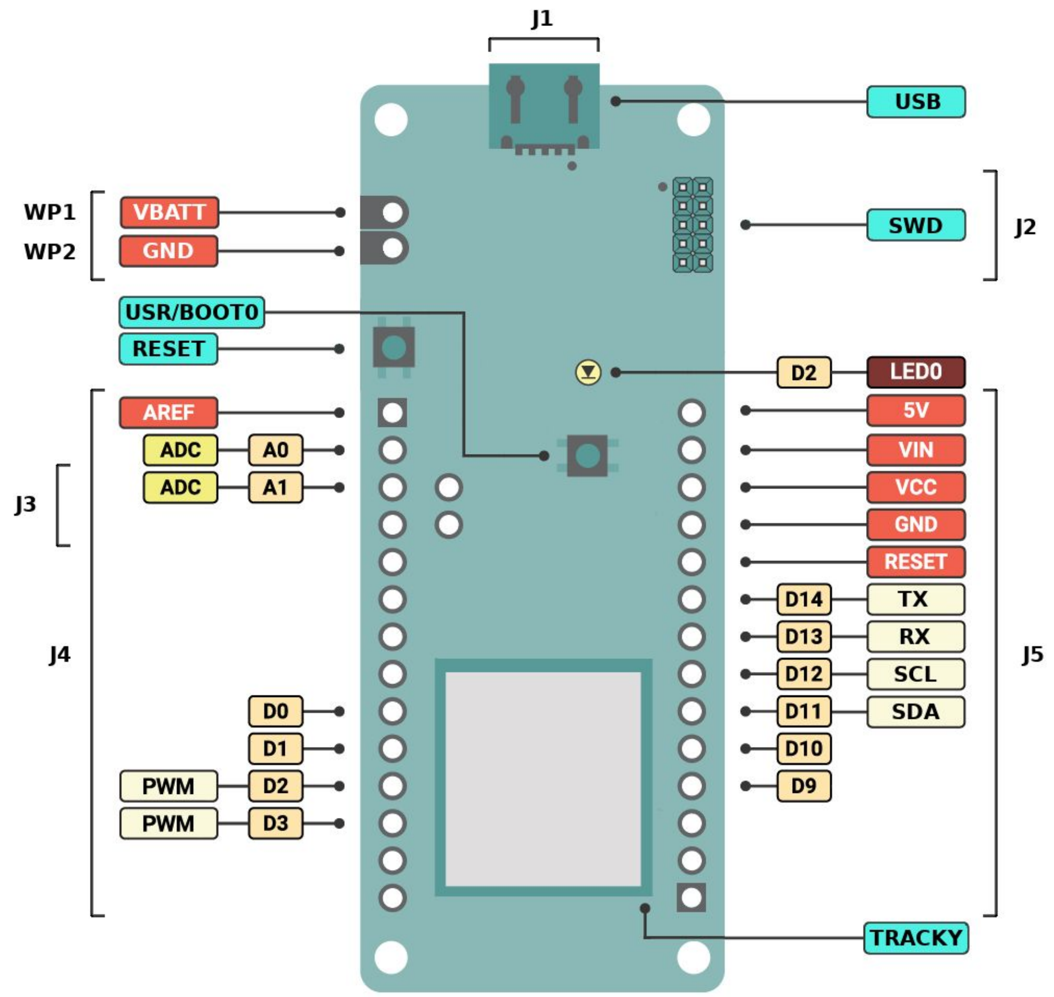

Figure 7. Tracky MKR board pinout

3.1 Arduino MKR Connectors

The connectors J4 and J5 provide the user with a standard Arduino MKR shield slot as listed below.

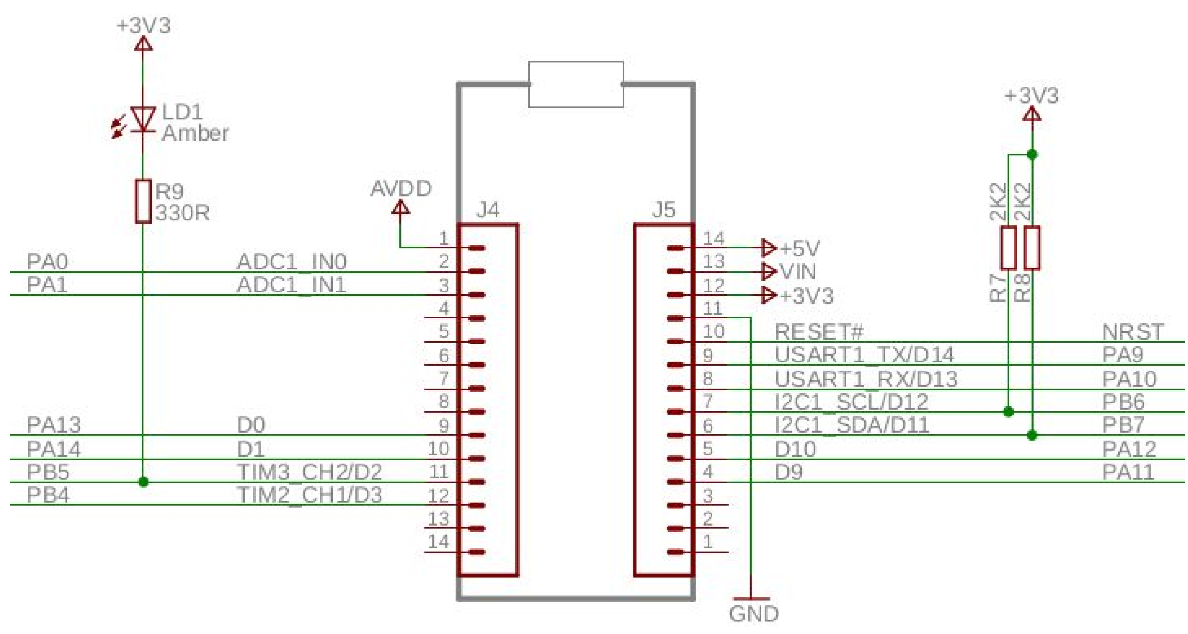

Figure 8. Tracky MKR Connectors

Conn	Arduino	STM	Description	Tracky Module pin
J4-1			VREF+	
J4-2	**A0**	**PA0**	ADC1_IN0/A0	
J4-3	**A1**	**PA1**	ADC1_IN1/A1	
J4-4	**A2**		N.C.	
J4-5	**A3**		N.C.	
J4-6	**A4**		N.C.	

J4-7	**A5**		N.C.	
J4-8	**A6**		N.C.	
J4-9	**D0**	**PA13**	D0	
J4-10	**D1**	**PA14**	D1	
J4-11	**D2**	**PB5**	TIM3_CH2/D2	
J4-12	**D3**	**PB4**	D3	
J4-13	**D4**		N.C.	
J4-14	**D5**		N.C.	
J5-1	**D6**		N.C.	
J5-2	**D7**		N.C.	
J5-3	**D8**		N.C.	
J5-4	**D9**	**PA11**	USB_DM/D9	
J5-5	**D10**	**PA12**	USB_DP/D10	
J5-6	**D11**	**PB9**	I2C1_SDA/D11	
J5-7	**D12**	**PB8**	I2C1_SCL/D12	
J5-8	**D13**	**PB7**	USART1_RX/D13	
J5-9	**D14**	**PB6**	USART1_TX/D14	
J5-10	**RESET**	7-NRST	RESET*	
J5-11	**GND**		GND	
J5-12	**VCC**		3V3	
J5-13	**VIN**		VIN	
J5-14	**5V**		5V	

Table 2: Tracky MKR pinout

* see reset circuit in DBG connector

3.2 J1 USB connector

The board is equipped with an USB (J1) Full-Speed (12 Mbps) device port on J1 connector. The Tracky MKR board can be powered through this interface.

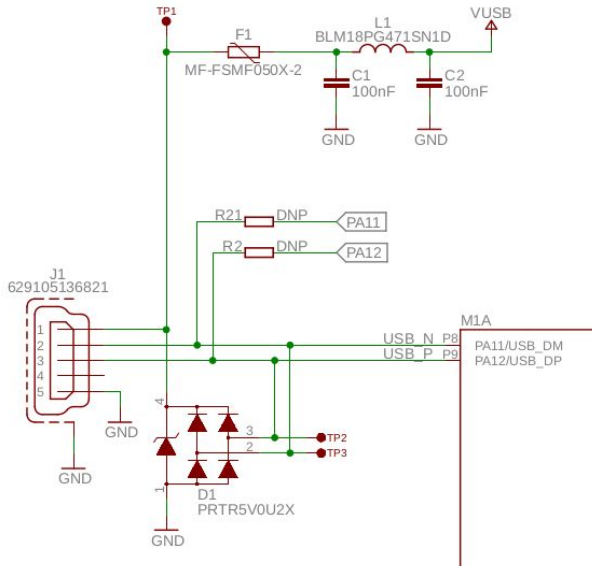

Figure 9. USB Interface connector

PA11 and PA12 pin can be optionally redirected to the J5-4 and J5-5 pins of Arduino MKR connector.

3.3 J2 SWD/Debug Connector

The Tracky MKR board features an on-board SWD Connector (J2) that can be used to program and debug the microcontroller.

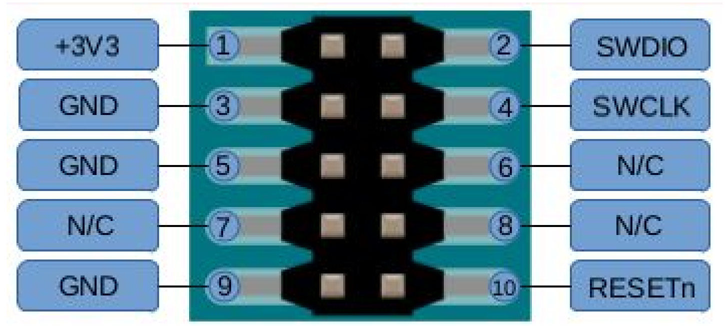

Figure 10. SWD Connector Pinout

Conn	STM	Description	Tracky Module pin
J2-1		3V3	
J2-2	**PA13**	JTSM-SWDIO	
J2-3		GND	
J2-4	**PA14**	JTCK-SWCLK	
J2-5		GND	
J2-6		NC	
J2-7		NC	
J2-8		NC	
J2-9		GND	
J2-10	**Pin 7**	NRST	

Table 3: SWD connector pinout

PA13 and PA14 are available on J4-9 and j4-10 on Arduino MKR connector

4 Usage

This chapter describes how to connect, configure and interact with the Tracky MKR board.

4.1 Power Supply

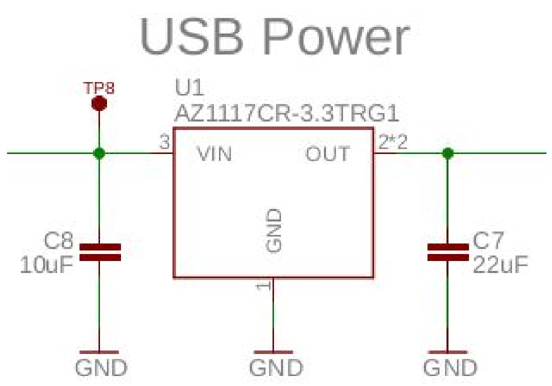

Figure 11. Tracky MKR Board Power Supply

Tracky MKR has an onboard AZ1117CR-3.3TRG1 300 mA low dropout linear regulator. The output voltage of the converter is 3.3V.

The board can be powered by three different power supplies sources:

- 5V on micro USB connector
- VBATT pin, from 3.3 V to 5.5 V.
- VIN pin, form 3.3V to 5.5V

4.3 Reset Button

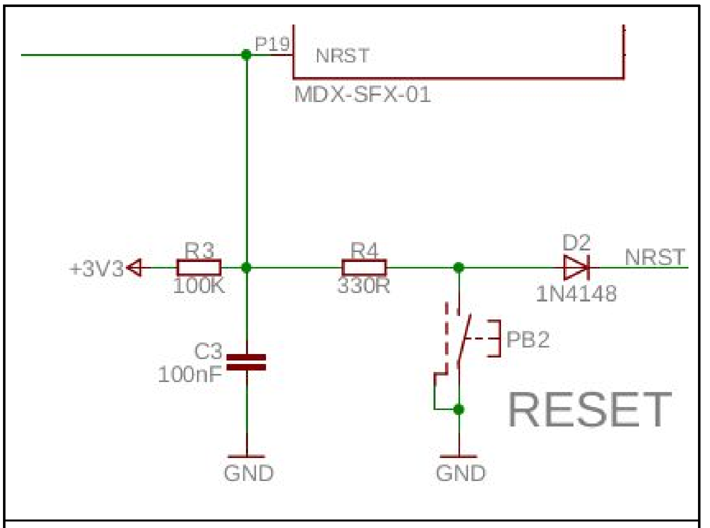

Figure 12. Reset button circuit

Push the button to reset the MCU

4.4 BOOT0 button

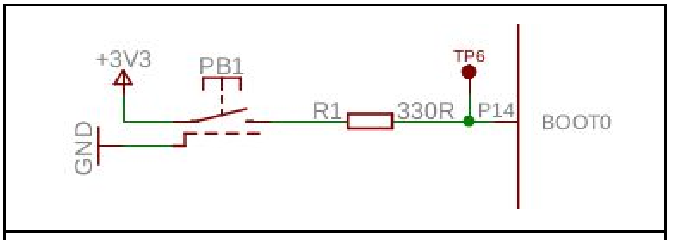

Figure 13. BOOT0 button circuit

The BOOT0 button can be pressed during reset to load the internal MCU bootloader.

4.5 LED

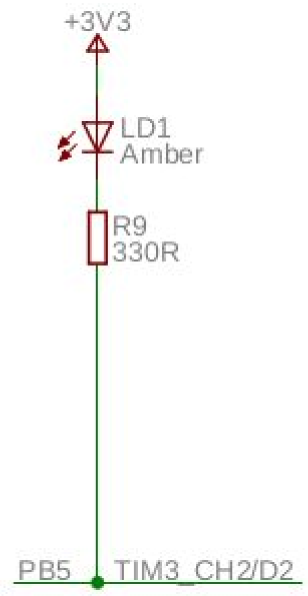

Figure 14. LED Circuit

The LD1 led is connected to the PE4 pin of the M4 core. Set the pin low to lit the LED.

5 Board Layout

The following pictures show the dimensions of the Tracky MKR Board.

6 Firmware Upload

The STM32L0 MCU inside the Tracky module has one M0+ core used for embedded communication stack and custom application.

The module is delivered with no firmware installed.

The GUI application for flashing firmware is STM32CubeProgrammer, available for Windows, Linux and MacOS operating systems. It can be downloaded from ST at:

https://www.st.com/en/development-tools/stm32cubeprog.html

The firmware for the MCU can be uploaded:

- Using an STLink V2 or V3 device connected to the SWD interface
- Using the embedded ROM Bootloader that is selected by rising the BOOT0 pin and setting nBOOT0 and nBOOT1 option bytes. In this case the firmware can be uploaded via USB or UART

6.1 Software Development

6.2 Atollic TrueStudio IDE

The firmware can be developed and uploaded with STLink V2 or V3 device using:

- STM32CubeMX v5.10 or superior code generator that can be downloaded from STMicroelectronics website at https://www.st.com/en/development-tools/stm32cubemx.html
- Atollic TrueSTUDIO for STM32 free compiler based on Eclipse IDE that can be downloaded from STMicroelectronics website at https://atollic.com/

The developed application runs on the M4 core and interfaces to the communication stack on M0+ core using the communication functions provided by ST.

IN STM32CubeMX tool select the STM32L072 MCU from MCU Selector.

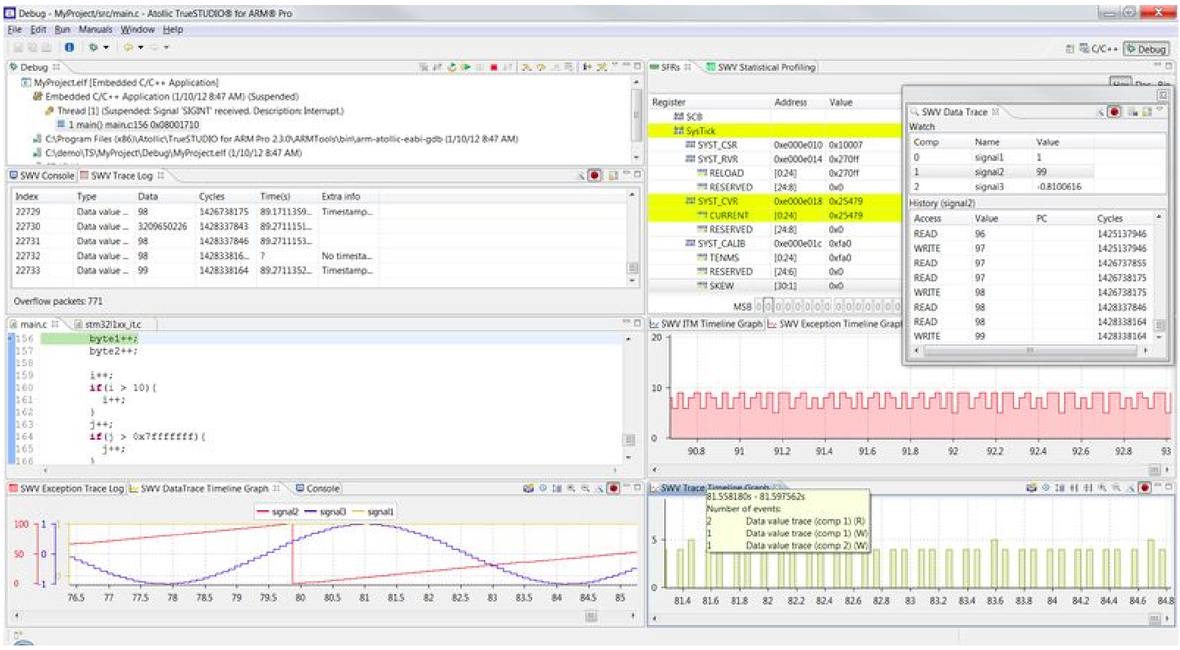

Figure 15. Atollic TrueStudio for STM32 IDE

In order to develop a custom firmware to be uploaded to the Tracky Module the following tools are necessary:

- A Windows/Linux/MacOS PC
- STM32CubeMX

- Atollic TrueSTUDIO for STM32
- STLink V2 or V3 device
  https://www.st.com/content/st_com/en/products/development-tools/hardware-development-tools/hardware-development-tools-for-stm32/st-link-v2.html

The ST-LINK/V2 is an in-circuit debugger and programmer for the STM8 and STM32 microcontroller families. The single wire interface module (SWIM) and JTAG/serial wire debugging (SWD) interfaces are used to communicate with any STM8 or STM32 microcontroller located on an application board.

6.3 Arduino IDE

7 References and Useful Links

7.1 Data Sheets and documents

- https://www.st.com/content/st_com/en/products/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus/stm32-ultra-low-power-mcus/stm32l0-series/stm32l0x2/stm32l072kz.html
- https://www.st.com/resource/en/datasheet/stm32l072kz.pdf
- https://www.st.com/resource/en/reference_manual/dm00108281.pdf

7.2 Tools

- https://www.st.com/content/st_com/en/products/microcontrollers/stm32-32-bit-arm-cortex-mcus/stm32-wireless-mcus/stm32wb-series/stm32wbx5/stm32wb55cg.html#sw-tools-scroll

7.3 WebSites

- http://www.midatronics.com
- https://www.st.com
- https://atollic.com/