Gwada-2

A big brain in a small body

Paul Bauerlé

Purpose of the document

This is a hobby documentation for my personal rocketry project. I am happy to share the content, and if this work can help other like I was helped by people sharing their project, it would be great.

Document the flight vehicle with a special attention to the avionics called BIBA.

Important documents

- BIBA modes flow charts
- Gwada-2
- Datasheets

Comments

ISSUE:

- No switch neither for powering on, nor to arm
  - not dramatic as no energetics is present BUT
  - BUT
  - consume more power as launch detection is directly on
  - SO
  - the idle mode will be a delay of 60s with led+buzzer blinking each 10s
  - Lesson learnt:
    - ALWAYS have an access to the avionics bay!
- Test to perform:
  - Current drawn in each mode
  - Battery voltage monitoring

Design procedure

1. Modes definition
2. Block diagram of the components and buses
3. FDIR
4. Power sizing
5. Component selection
6. Electrical schematic
7. Electrical Layout
8. Coding
9. FDIR
10. Component Validation
11. Breadboard wiring
12. Testing
13. FDIR
14. Third party validation
15. Protoboard wiring
16. Testing
17. FDIR
18. Third Party Validation
19. FDIR

Development program

- Mission Analysis
  - Flight Simulation
  - Modes definition
  - Avionics functions
- Hardware
  - Design
  - Wiring and testing
- Software
  - Data structure
  - Unit coding
  - Flight software coding
- Operations

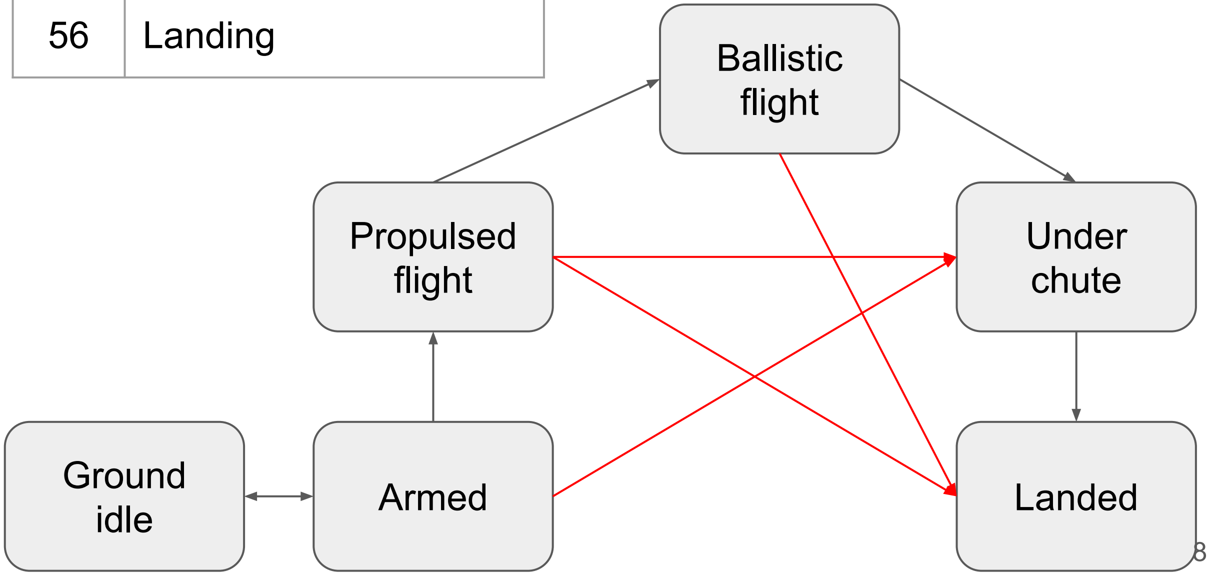
Mission Analysis

States transitions

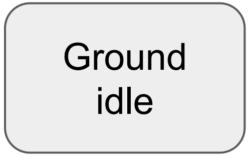
ID	State
1	Ground idle
2	Armed
3	Propulsed flight
4	Ballistic flight
5	Under chute
6	Landed

ID	Transition
121	Arming switch
23	Lift-off
34	Burnout
45	Chute deployment
56	Landing

ID	Transition
24	Chute deployment
35	Chute deployment
36	Hot landing
46	Ballistic landing



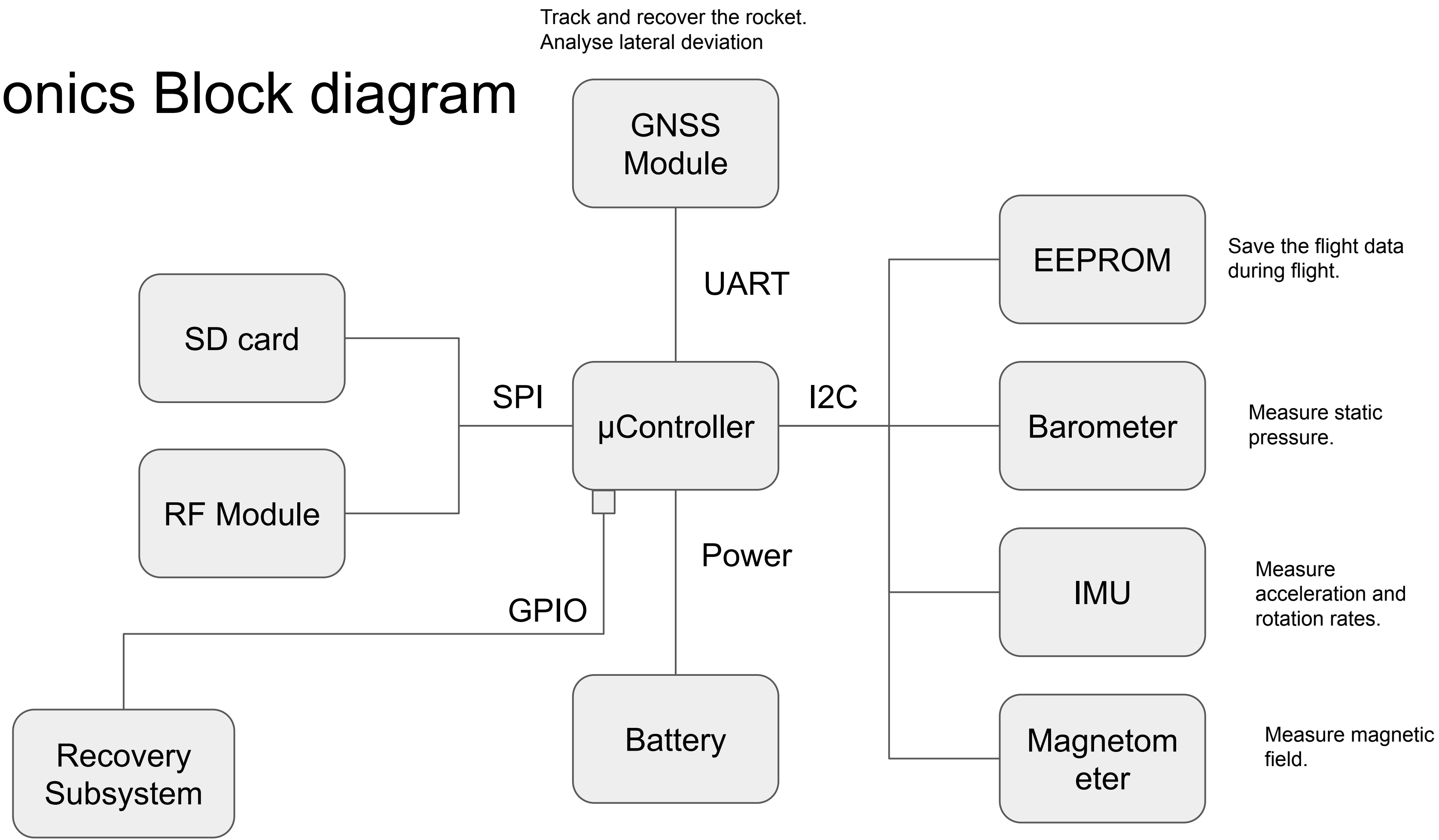
Mode description: Ground Idle



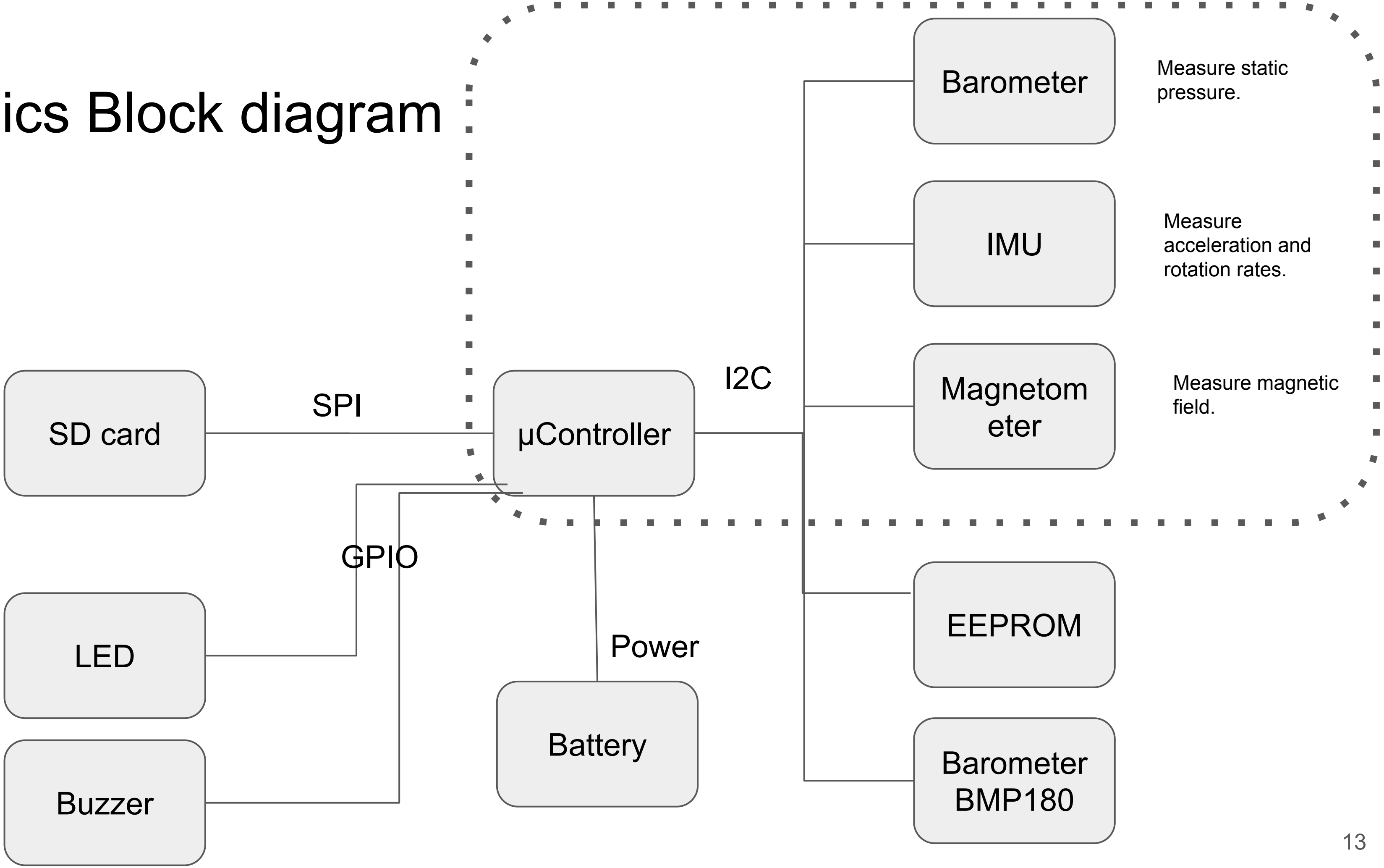
ID	State	Description
1	Ground idle	Low consumption. No power channel can be armed. Sensors can be tested.
2	Armed	Launch detection method: Axial Acceleration for a period of time to define. Mean pressure determination. Circular memory for data savin. (Save everything for the first flights)
3	Propulsed flight	High measurement rate (around 10 Hz TBTested)(depends on memory) Save on EEPROM all data. Send on Telemetry all data. Detect burnout by acceleration drop.
4	Ballistic flight	High measurement rate (around 10 Hz TBTested)(depends on memory) Save on EEPROM all data. Send on Telemetry all data. Detect apogee by vertical descent for a period of time. Deploy chute after a flight duration greater than simulated time to apogee+dt.
5	Under chute	High/low measurement rate (around 10 Hz TBTested)(depends on memory) Save on EEPROM all data. Send on Telemetry all data. Detect landing: under 10m AGL for 10 s, and/or no altitude change after 10s,or flight time significantly greater than predicted.
6	Landed	Copy data from EEPROM and paste on SD card. Buzz for rocket tracking. Save power.

Hardware

Avionics Block diagram



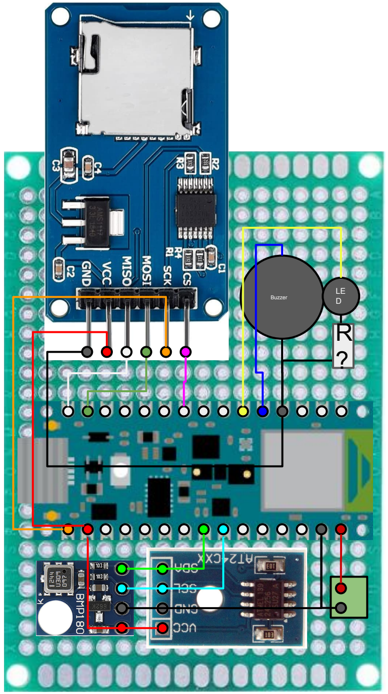
Avionics Block diagram



Wiring and testing

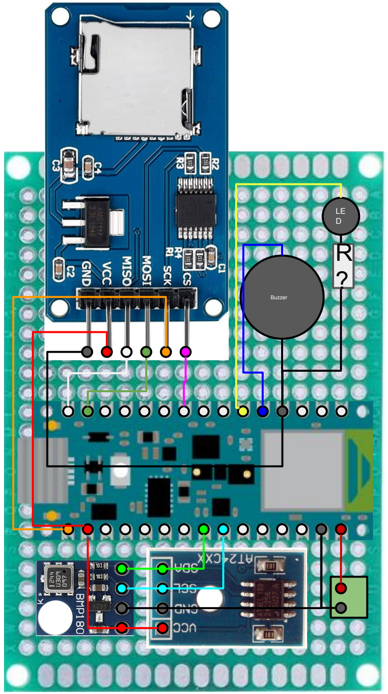
The avionics board will be a protoboard with soldered female pin headers to

Protoboard wiring layout



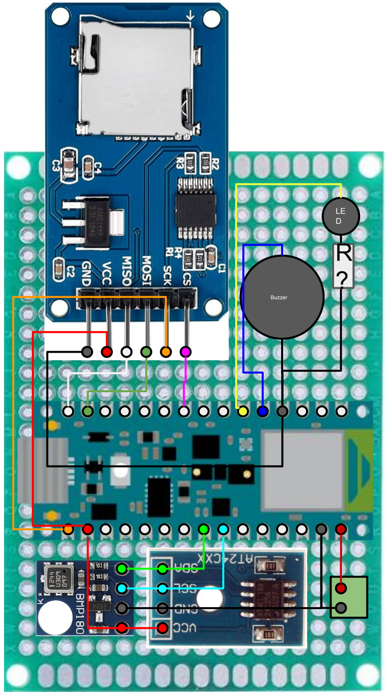
3.3V	Periph Vcc
GND	Ground
SDA	I²C
SCL	I²C
MISO	SPI
MOSI	SPI
SCK	SPI
D6	Slave Select SD Reader
D2	Buzzer
D3	Led

Arduino Pinout connexions



EEPROM	I²C	VCC	3.3V
		GND	GND
		SCL	SCL
		SDA	SDA
External Baro	I²C	VCC	3.3V
		GND	GND
		SCL	SCL
		SDA	SDA
SD reader	SPI	CS	D6
		SCK	SCK
		MOSI	MOSI
		MISO	MISO
		VCC	3.3V
		GND	GND
Buzzer	Digital	+	D2
Led	Digital	+	D3
Battery	Power	Bat+	Vin
		Bat-	GND

Breadboard wiring layout



insert pictures

3.3V	Periph Vcc
GND	Ground
SDA	I²C
L	I²C
O	SPI
SI	SPI
K	SPI
	Slave Select SD Reader
D2	Buzzer
D3	Led

Using the SD card reader on 3.3V

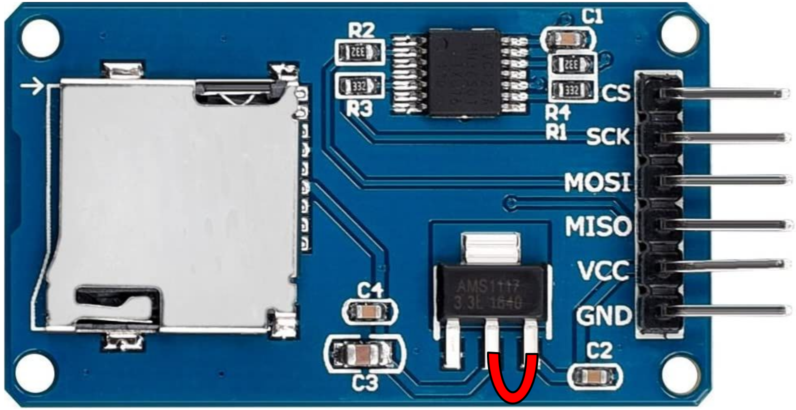
The SD card reader did not work on 3.3V but worked using the 5V with the Uno.

Even if the supplier wrote it allows both here, it does not look like it.

I saw on a forum that the voltage regulator could be shorted to support the 3.3V.

It was mentioned with no guarantee!!

I tried, it worked!



Bridge here
at your risks…

Software

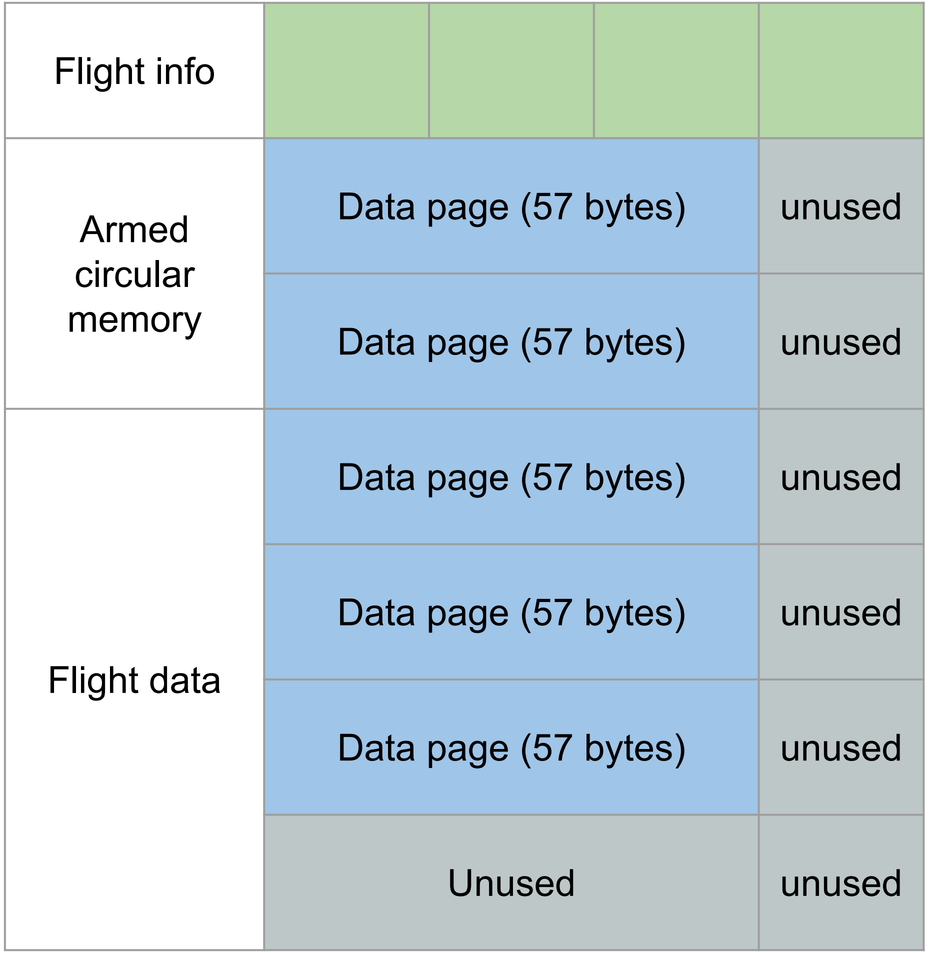
Data structure

Description	Unit	Type	Size (bytes)	Start position	Min val	Max val
Time	ms	float (or int)	4	0		
Pressure	kPa	float	4	4		
Altitude	m	float	4	8		
Vertical velocity	m	float	4	12		
Acceleration X	m/s^2	float	4	16		
Acceleration Y	m/s^2	float	4	20		
Acceleration Z	m/s^2	float	4	24		
Rotation rate X	deg/s /!\	float	4	28		
Rotation rate Y	deg/s /!\	float	4	32		
Rotation rate Z	deg/s /!\	float	4	36		
x-Magnetic field	µT	float	4	40		
y-Magnetic field	µT	float	4	44		
z-Magnetic field	µT	float	4	48		
Temperature	°C /!\	float	4	52		
Pressure_bmp	kPa	float	4	56		
Mode	-	byte	1	60		
		Total	**61**			

Details on the sheet [Gwada-2/Data structure]

61 Bytes on each data pack

Allow page writing (5ms for 64B))



Flash memory / EEPROM Organisation

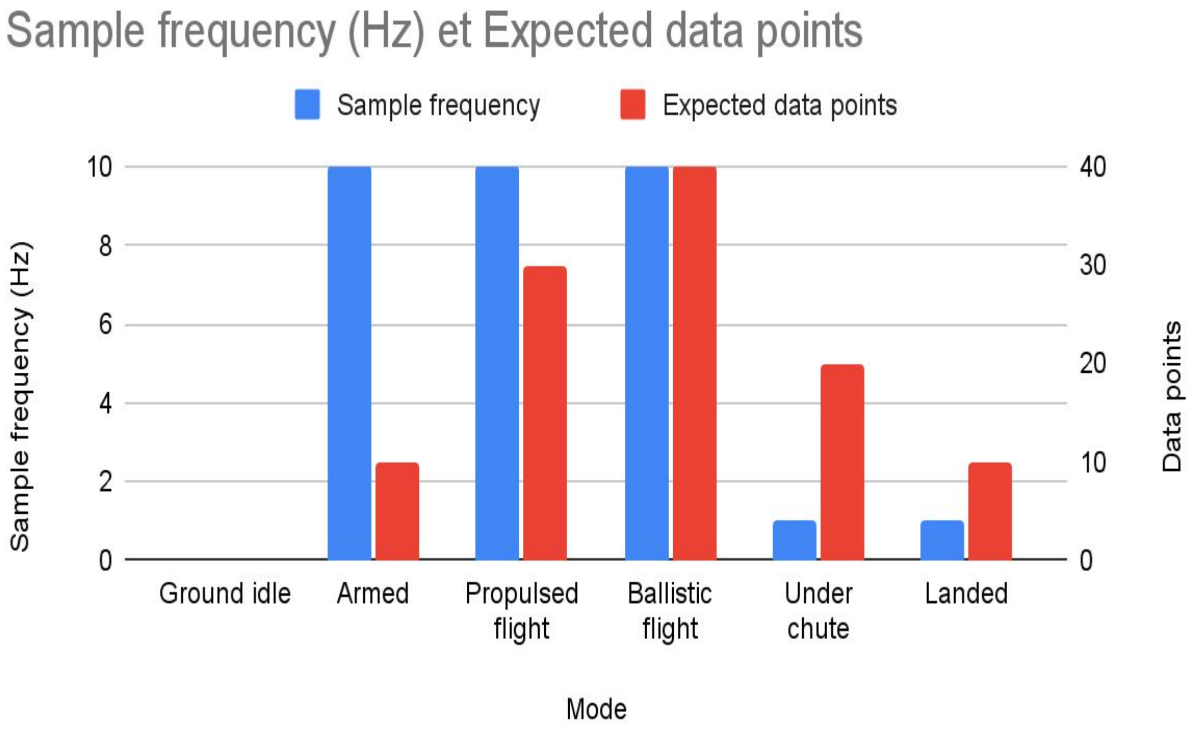
EEPROM datasheet

- The 256K is internally organized as **512 pages of 64-bytes each.**
- **2 page** for **flight info**
- **50 pages** to save the data in **armed mode**, continuously saving on a circular way
- **460 pages** for the **flight data**

EEPROM & I2C limitations

- EEPROM write cycle is 5ms
- can perform page write in 5ms (1 page=64bytes)
- BUT the buffer to send the data over I2C is apparently limited to 32 bytes
  - this is due to the wire library,
- At the moment, I can send up to 4 floats (16Bytes)
  - TO FIX: I WANT AT LEAST 32 BYTES!
- Solutions:
  - Modify the size of the buffer in the wire library: documentation
  - **Send the data in 4 operations**
    - pro: simple, easy to implement
    - cons: 20ms instead of 5ms, can decrease to 10ms if 32B can be sent
  - use another library/code manually the I²C communication
    - pro: only 5ms to send 64B
    - cons: need to search for a lib or code/modify + tests, last longer

Sample frequency



Using circular memory protocol one 50 pages (5s data) while system is armed and launch not detected.

Total expected points:

~110 < 460

No memory issue

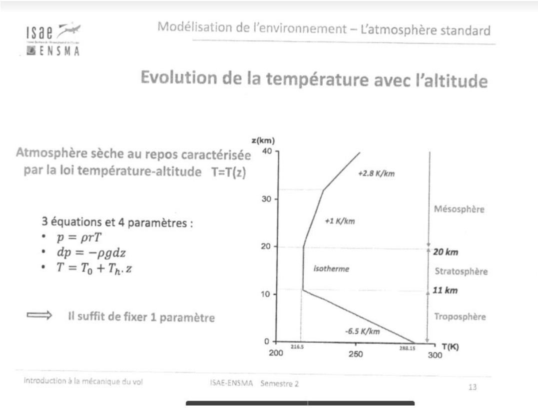
Altitude as function of air pressure

Initial equations

$$\frac{dP}{dz} = -\rho g$$

$$P = \rho r T$$

$$T = T_0 - z \cdot K_T$$



Some derivation using the equations

$$\frac{dP}{dz} = -\frac{P}{rT} g$$

$$\frac{dP}{P} = -\frac{g}{rT} dz$$

$$\frac{dP}{P} = -\frac{g}{r} \frac{dz}{T_0 - z \cdot K_T}$$

$$\int_{P_0}^{P} \frac{dP}{P} = -\frac{g}{r} \int_{z_0}^{z} \frac{dz}{T_0 - z \cdot K_T}$$

$$[\ln P]_{P_0}^{P} = +\frac{g}{r} \left[\frac{\ln(T_0 - z \cdot K_T)}{K_T} \right]_{z_0}^{z}$$

$$\ln\left(\frac{P}{P_0}\right) = +\frac{g}{r K_T} \ln\left(\frac{T_0 - z \cdot K_T}{T_0 - z_0 \cdot K_T}\right)$$

Altitude as function of air pressure

$$\ln\left(\frac{P}{P_0}\right) = +\frac{g}{rK_T}\ln\left(\frac{T_0 - z\cdot K_T}{T_0 - z_0\cdot K_T}\right)$$

$$\ln\left(\frac{P}{P_0}\right) = ln\left(\left(\frac{T_0 - z\cdot K_T}{T_0 - z_0\cdot K_T}\right)^{+\frac{g}{rK_T}}\right)$$

$$\frac{P}{P_0} = \left(\frac{T_0 - z\cdot K_T}{T_0 - z_0\cdot K_T}\right)^{\frac{g}{rK_T}}$$

$$\left(\frac{P}{P_0}\right)^{\frac{rK_T}{g}} = \frac{T_0 - z\cdot K_T}{T_0}$$

$$\left(\frac{P}{P_0}\right)^{\frac{rK_T}{g}} = 1 - z\cdot\frac{K_T}{T_0}$$

$$z\cdot\frac{K_T}{T_0} = 1 - \left(\frac{P}{P_0}\right)^{\frac{rK_T}{g}}$$

With

$$K_T = -6.5^\circ C/km = -0.0065\,K/m$$
$$g = 9.81\,m/s^2$$
$$r = 287\,J/K/kg$$

Numerically, the altitude z Above Ground Level in meter is:

$$\boxed{z = \frac{T_0}{0.0065}\left[1 - \left(\frac{P}{P_0}\right)^{\frac{1}{5.25864}}\right]}$$

with $\mathbf{T_0}$ **in K**!

Altitude as function of air pressure

Isolating the altitude z.

As we are interested in Altitude Above Ground Level (AGL), the origin is at ground level, so z_0=0.

Numerically, the altitude z Above Ground Level in meter is:

$$\boxed{z = \frac{T_0}{0.0065}\left[1 - \left(\frac{P}{P_0}\right)^{\frac{1}{5.25864}}\right]}$$

with **T_0 in K**!

It **differs from formula** we can find on the web:

Hypsometric formula

$$h = \frac{\left(\left(\frac{P_0}{P}\right)^{\frac{1}{5.257}} - 1\right) \times (T + 273.15)}{0.0065}$$

Source:https://keisan.casio.com/exec/system/1224585971

BUT equivalent from the one seen in my flight mechanics course :)

Troposphère: 0 à 11km

$p/p_0 = (1 - 0.02256\ z)^{5.259}$

$\rho/\rho_0 = (1 - 0.02256\ z)^{4.259}$

avec:

z en km

p_0 = 101325 Pa

ρ_0 = 1.225 kg m^{-3}

T_0 = 288.15 K

Ground level pressure and temperature

- When the system is armed:
  - each 10 s, measure 5 sample of Pressure & Temperature
  - Average values as followed:

$$P_0 = \frac{P_{0,new\,meas} + n_{old} \cdot P_{0,old}}{n_{old} + 1}$$

$$T_0 = \frac{T_{0,new\,meas} + n_{old} \cdot T_{0,old}}{n_{old} + 1}$$

with

P_0 : Averaged ground level pressure

T_0 : Averaged ground level temperature

$P_{0,old}$: Previous value of P_0

$T_{0,old}$: Previous value of T_0

n_{old} : Number of previous data points

$P_{0,new\,meas}$: Average of the 5 new measurements of Pressure

$T_{0,new\,meas}$: Average of the 5 new measurements of Temperature

This method allows to filter wind and noises while adapting to the temperature variation if the rocket stay a long time on the launch pad.

Arduino Nano 33 BLE Sense

Cheat sheet

Barometric sensor:

H = 44330 * [1 - (P/p0)^(1/5.255)]

Where, "**H**" stands for altitude, "**P**" the measured pressure (kPa) from the sensor and "**p0**" is the reference pressure at sea level (101.325 kPa).

Operations

Appendices

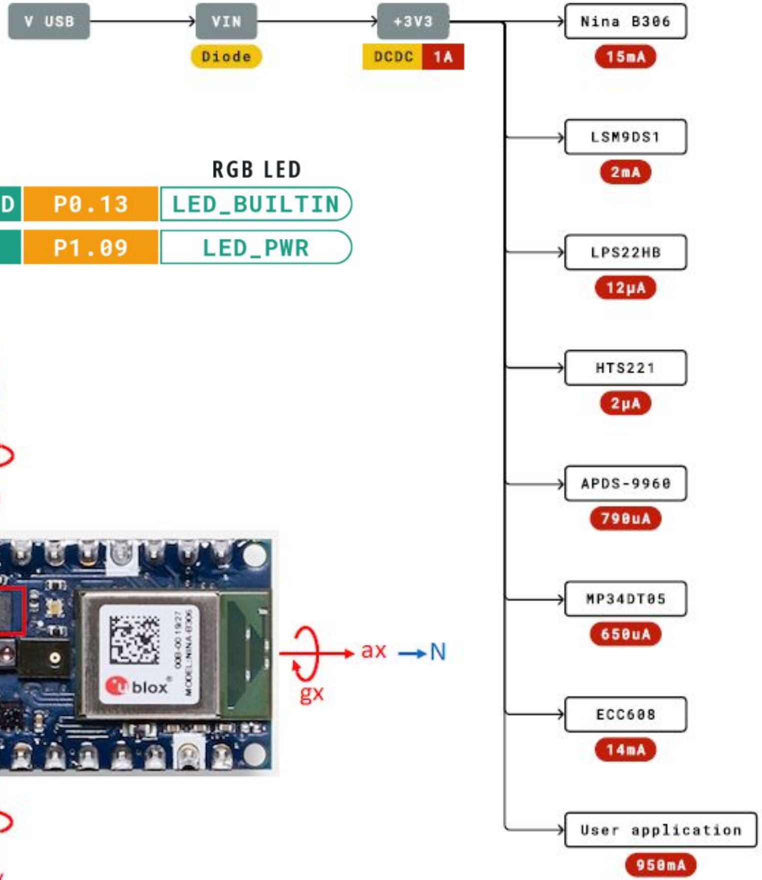
Archives

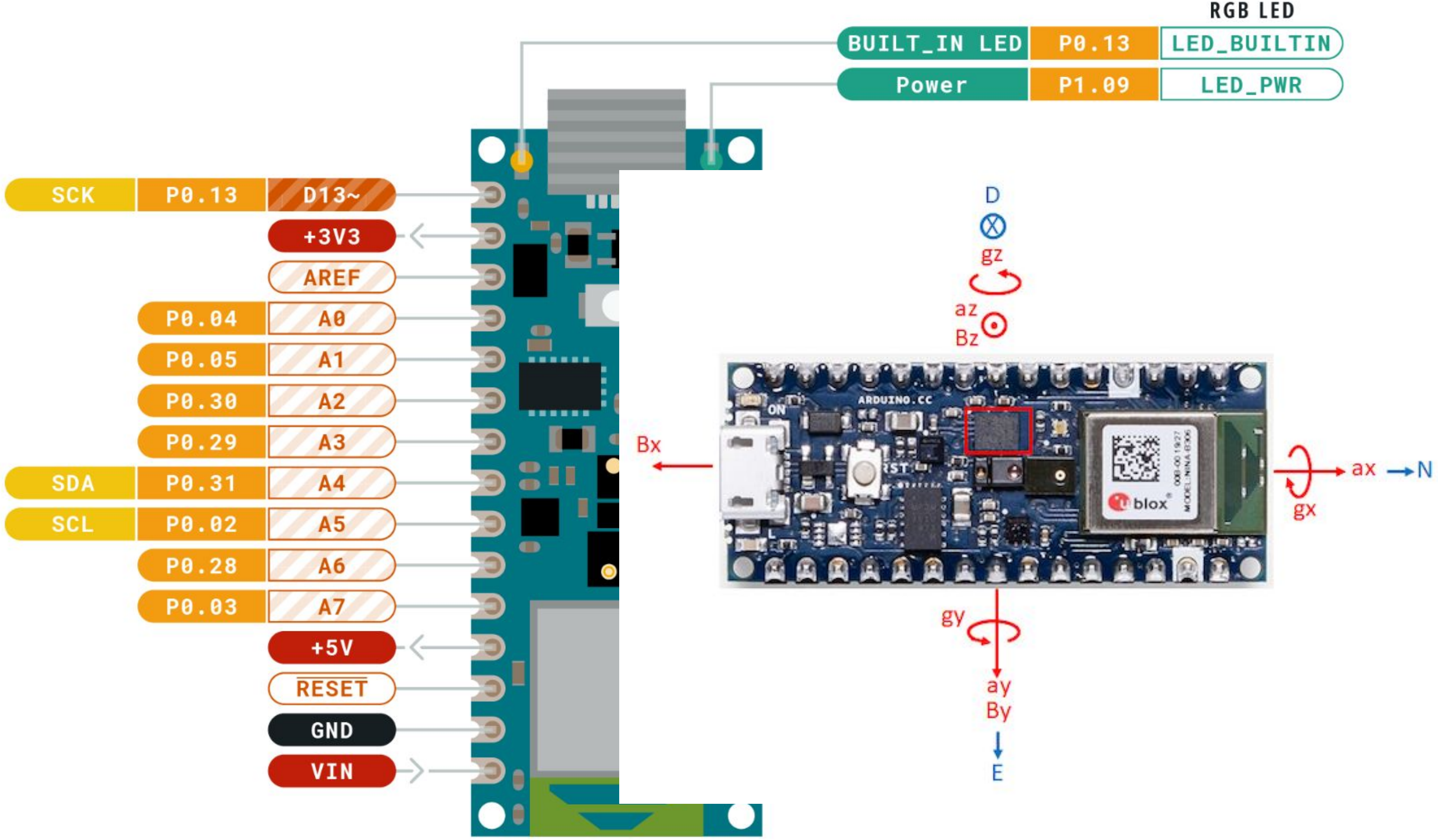
Power budget

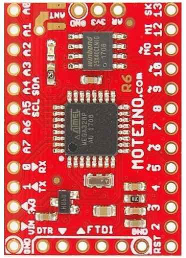
Arduino Nano 33 BLE Sense	20 mA (TBC)
IMU (LSM9DS1TR)	2 mA
Barometer (LPS22HB)	12 µA
DC Current per I/O Pin	10 mA
Total continuous consumption	25 mA

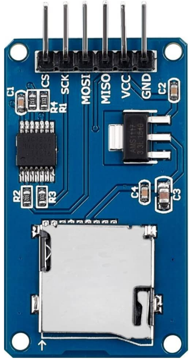
Battery Capacity	300 mAh
Battery duration	12 h

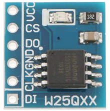
Arduino Nano 33 BLE Sens

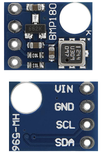


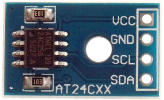


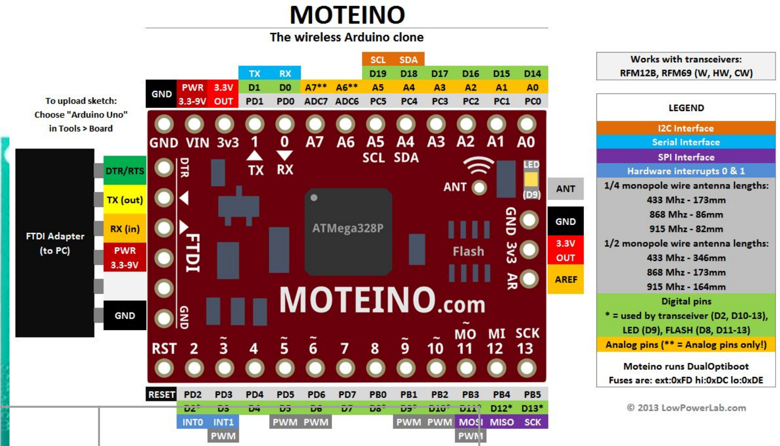




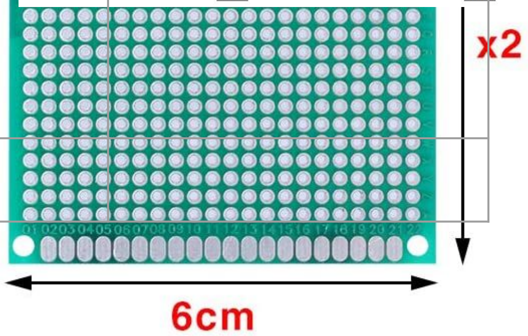


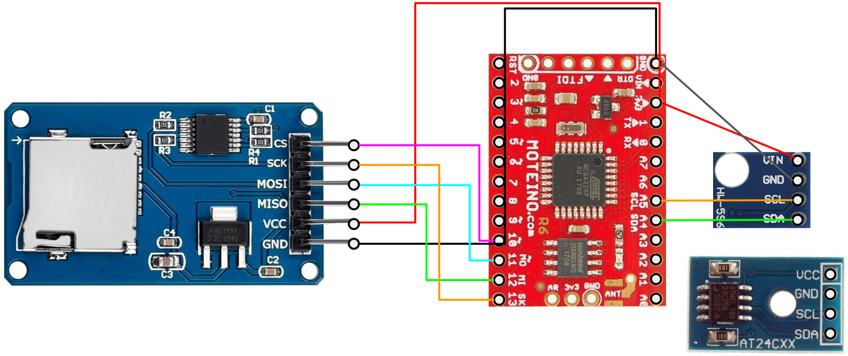


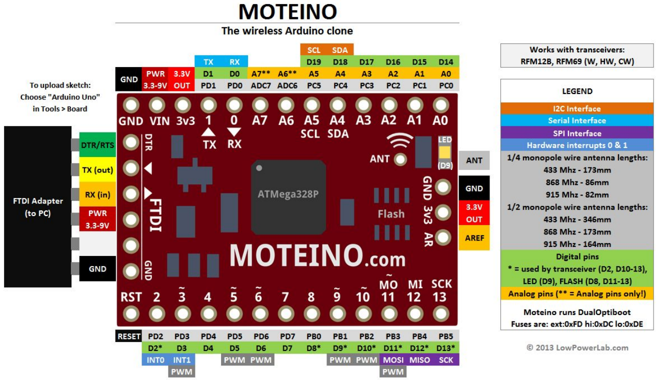


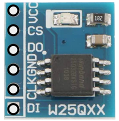


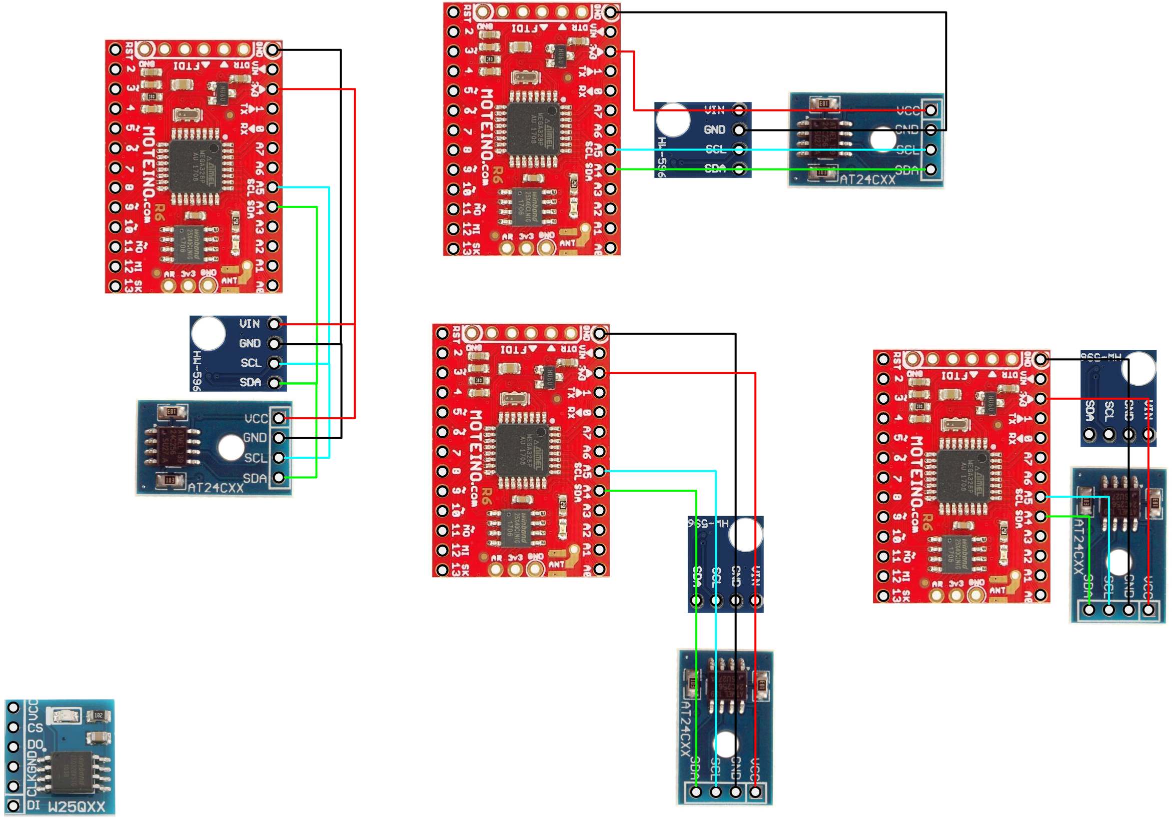
µSD reader SPI	Barometer, BMP180 S2C	EEPROM S2C AT24C256

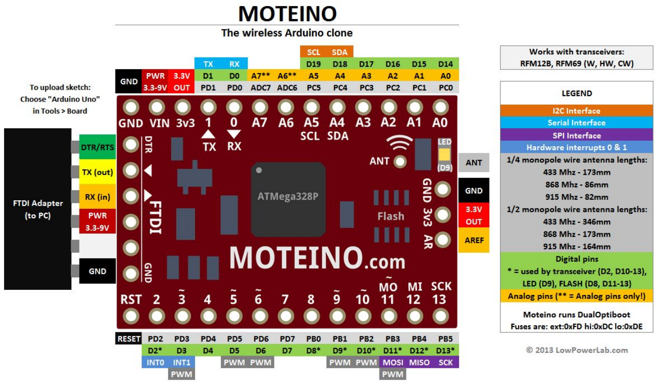


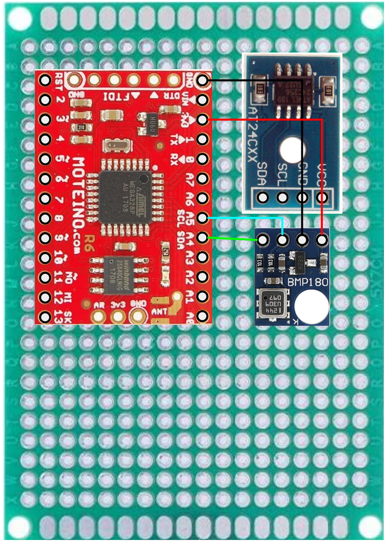

CS
SCK
MOSI
MISO
VCC
GND
MOTEINO.com
VIN
GND
SCL
SDA
VCC
GND
SCL
SDA
AT24CXX

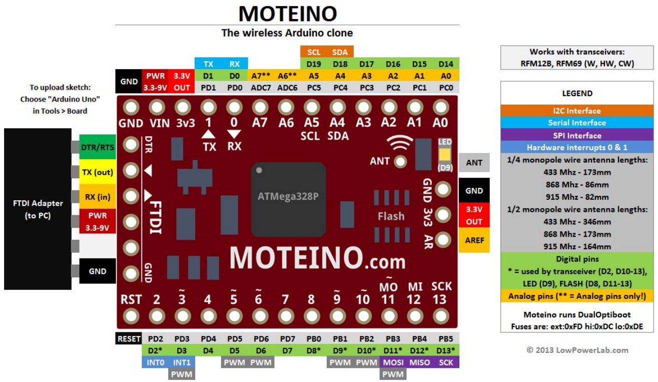

MOTEINO
The wireless Arduino clone
To upload sketch:
Choose "Arduino Uno"
in Tools > Board
FTDI Adapter
(to PC)
DTR/RTS
TX (out)
RX (in)
PWR
3.3-9V
GND
ATMega328P
Flash
MOTEINO.com
Works with transceivers:
RFM12B, RFM69 (W, HW, CW)
LEGEND
I2C Interface
Serial Interface
SPI Interface
Hardware interrupts 0 & 1
1/4 monopole wire antenna lengths:
433 Mhz - 173mm
868 Mhz - 86mm
915 Mhz - 82mm
1/2 monopole wire antenna lengths:
433 Mhz - 346mm
868 Mhz - 173mm
915 Mhz - 164mm
Digital pins
* = used by transceiver (D2, D10-13),
LED (D9), FLASH (D8, D11-13)
Analog pins (** = Analog pins only!)
Moteino runs DualOptiboot
Fuses are: ext:0xFD hi:0xDC lo:0xDE
© 2013 LowPowerLab.com


VCC
CS
DO
GND
CLK
DI
W25QXX




MOTEINO
The wireless Arduino clone
To upload sketch: Choose "Arduino Uno" in Tools > Board
FTDI Adapter (to PC)
DTR/RTS
TX (out)
RX (in)
PWR 3.3-9V
GND
Works with transceivers: RFM12B, RFM69 (W, HW, CW)
LEGEND
I2C Interface
Serial Interface
SPI Interface
Hardware interrupts 0 & 1
1/4 monopole wire antenna lengths:
433 Mhz - 173mm
868 Mhz - 86mm
915 Mhz - 82mm
1/2 monopole wire antenna lengths:
433 Mhz - 346mm
868 Mhz - 173mm
915 Mhz - 164mm
Digital pins
* = used by transceiver (D2, D10-13), LED (D9), FLASH (D8, D11-13)
Analog pins (** = Analog pins only!)
Moteino runs DualOptiboot
Fuses are: ext:0xFD hi:0xDC lo:0xDE
© 2013 LowPowerLab.com


FTDI
MOTEINO.com
BMP180


MOTEINO
The wireless Arduino clone
To upload sketch:
Choose "Arduino Uno"
in Tools > Board
FTDI Adapter
(to PC)
DTR/RTS
TX (out)
RX (in)
PWR
3.3-9V
GND
ATMega328P
Flash
ANT
MOTEINO.com
Works with transceivers:
RFM12B, RFM69 (W, HW, CW)
LEGEND
I2C Interface
Serial Interface
SPI Interface
Hardware interrupts 0 & 1
1/4 monopole wire antenna lengths:
433 Mhz - 173mm
868 Mhz - 86mm
915 Mhz - 82mm
1/2 monopole wire antenna lengths:
433 Mhz - 346mm
868 Mhz - 173mm
915 Mhz - 164mm
Digital pins
* = used by transceiver (D2, D10-13),
LED (D9), FLASH (D8, D11-13)
Analog pins (** = Analog pins only!)
Moteino runs DualOptiboot
Fuses are: ext:0xFD hi:0xDC lo:0xDE
© 2013 LowPowerLab.com