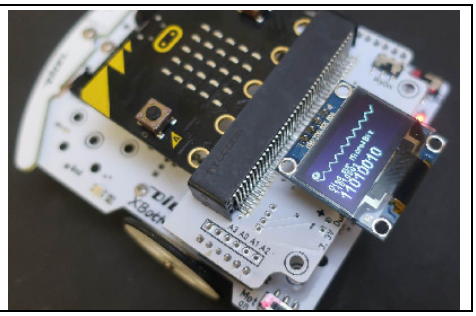
Micro:Bit – C programming under Arduino IDE

The BBC Micro:Bit for education is mostly programmed with Blockly, Blocks and derived names. MicroPython, TigerJyton, JavaScript are the other options. Arduino offers now a quite interesting option for connecting I/O devices.	 

Arduino programming

Adafruit document how to start and load examples
https://learn.adafruit.com/use-micro-bit-with-arduino

There is a more technical documentation under
https://cdn-learn.adafruit.com/downloads/pdf/use-micro-bit-with-arduino.pdf

See how to install Micro:Bit emulation on
https://learn.adafruit.com/use-micro-bit-with-arduino/led-matrix
and load the software from
https://sandeepmistry.github.io/arduino-nRF5/package_nRF5_boards_index.json

When everything is installed, select BBC micro:bit under tools. Check if Port is recognized.

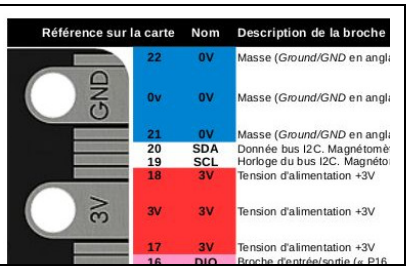
Our approach here provides a deeper understanding of the Micro:Bit hardware

Micro:Bit pins and local devices

What is not so well documented is the pin number to use for selecting the pins of the Micro:Bit connector and how to access signals that are not on the connector. When developing a card, the designer works with the pin numbers of the microcomputer chip. These pins are connected to the I/O ports the system programmer is working with, at the lowest level. On the nRF51822 Microcontroller hart of the Micro:Bit, there is a single 32-bit register, with bits numbered 0 to 31. Now the Micro:Bit board is designed. A 40-pin connector is used and these pins are numbered 1 to 40. But several pins are connected together and numbers and names are given to the signals. Arduino software works with pin numbers which are not the same. Easy to get confused!
On the Micro:Bit schematic and connector pinout, the Arduino pin numbers does not appear. The important info listed in the table below is

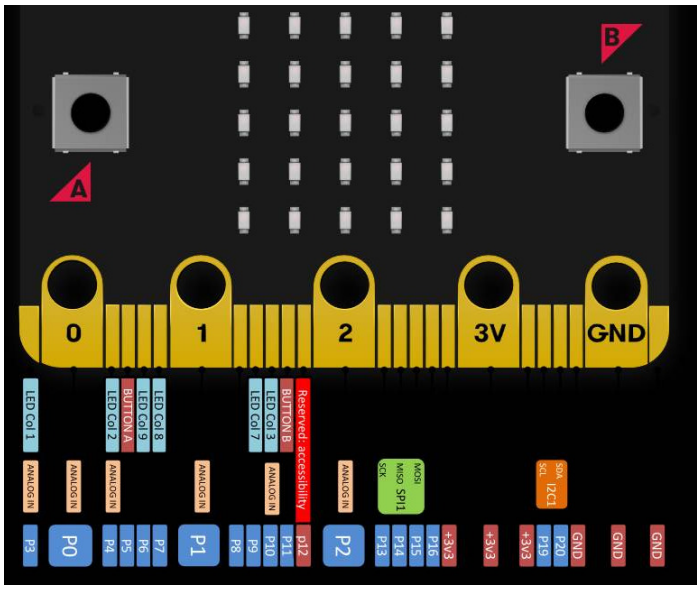
- the bit number on the nRF51 processor; this will be useful to work at the lowest level
- the "Arduino" pin number to be used with pinMode and digitalWrite
- the name given on the Micro:Bit connector.

Name		Arduino pin def	Nrf51 register
P0 Ana		A0	P0.03
P1 Ana		A1	P0.02
P2 Ana		A2	P0.01
P3 Ana	Col1	3 - A3	P0.04
P4 Ana	Col2	4 - A4	P0.05
P10 Ana?	Col3	10	P0.06



P9-10	(Col4)	23 ?10	P0.07	
P9-10	(Col5)	24 ?10	P0.08	
P9-10	(Col6)	25	P0.09	
P9-10	Col7	9	P0.10	
P7	Col8	7	P0.11	
P6	Col9	6	P0.12	
	(Row1)	26	P0.13	
	(Row2)	27	P0.14	
	(Row3)	28	P0.15	
BTN_A	ButtonA	5	P0.17	
BTN_B	ButtonB	11	P0.26	
SCL	I2C Scl	19	P0.00	
SDA	I2C Sda	20	P0.30	
SCK	SPI Sck	13	P0.23	
MISO	SPI MISO	14	P0.22	
MOSI	SPI MOSI	15	P0.21	
DIO P18	DIO	16	P0.20?	
DIO P8	DIO	12	P0.16?	
	(Rx)		P0.25	
	(Tx)		P0.24	

On the board, the 25 Leds are connected in a strange way. One need to have Ri High and Cj Low to light the Led at position Ri/Cj



Surprisingly, if you blink Column 4,5,6, you see P9 and P10 contacts that blinks. But Leds are correctly selected according to table above.

The following test program blinks any Led; one need to set the row and the column, which is special due to not documented design and/or PCB layout constraints.

```
// Blink one Led
//MicroBit  1752b
#define R1 26
#define R2 27
#define R3 28
#define C1 3
#define C2 4
#define C3 10
#define C4 23
#define C5 24
#define C6 25
#define C7 9
#define C8 7
#define C9 6
```

```
void setup() {
  pinMode(Col, OUTPUT);
  pinMode(Row, OUTPUT);
}
void SetLiCo (byte li,byte
co) {
  digitalWrite(li, HIGH);
  digitalWrite(co, LOW);
}
void ClrLiCo (byte li,byte
co) {
  digitalWrite(li, HIGH);
digitalWrite(co, HIGH);
}
```

```
//Select Row/Col that
blinks
#define Row R3
#define Col C3

void loop() {
  SetLiCo(Row,Col);
delay(200);
  ClrLiCo(Row,Col);
delay(300);
}
```

Coding is easy, but not elegant.

Doing the same on the Arduino C-compiler using the nrf51description looks like this:

```
// Blink one Led   - direct registers access
#include <nrf.h>
#define LED 13

#define COL1  4
#define ROW1  13
void setup() {
  NRF_GPIO->DIRSET = 1<<ROW1;
  NRF_GPIO->DIRSET = 1<<COL1;
  NRF_GPIO->OUTSET = 1<<ROW1;
  NRF_GPIO->OUTCLR = 1<<COL1;
//  pinMode(LED, OUTPUT);
}
```

```
void RTC1_IRQHandler(void) {
}

void loop() {
  for(;;) {
  NRF_GPIO->OUTSET = 1<<COL1;
//  digitalWrite(LED, HIGH);
//  delay(200);
                  // wait ?
//  digitalWrite(LED, LOW);    // turn the LED off by making the
voltage LOW
NRF_GPIO->OUTCLR = 1<<COL1;
```

Working with set of bits is of course more compact and efficient. Feedback if you can help.

Timing comparison

Blink period without delay is 0.5 microsecond using nRF51 and the C-compiler of Arduino. With Arduino functions it takes 4 microseconds.
If the same is done on the AVR328, 16MHz also, one gets 0.4 microsecond using bitSet/bitClear on a register bit and 7 microseconds with Arduino digitalWrites .

Future work

Github didel libx set of inserted files is efficient, but cannot be used as such on the Micro:Bit for the moment. Definition files must be adapted to use Arduino I/O functions and a timer interrupt is required.

The libraries that have been adapted on Micro:Bit are
  OledMicrobit.h Oled SSD1306 on any 2 pins
  Apa102Min.h Apa102/Sk9822 strip on any 2 pins
  BbI2C.h I2C on any 2 pins
  DHT22.h read temp/humidity on any pin

  Contact info@didel.com if interested.

Jdn 191209