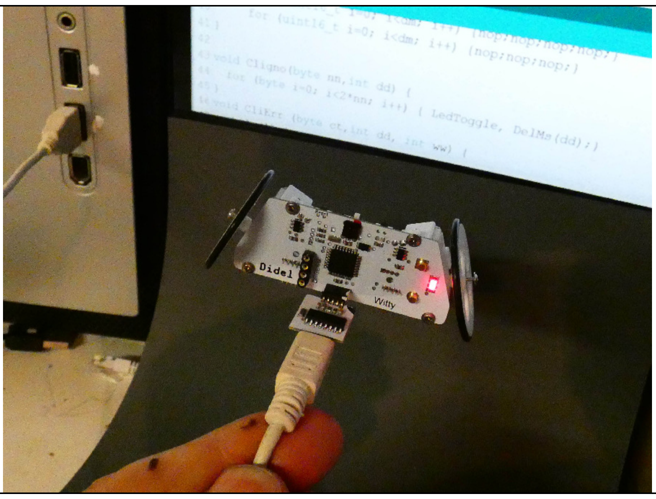
Witty - software

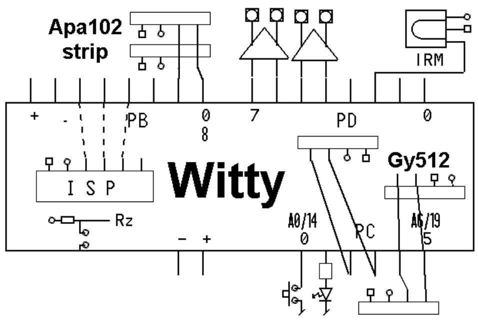
The Witty is a 2-wheel robot with 2 stable states. Rolling behavior is surprising and interesting to control.
An RGB strip brings animation and offer plenty of programming possibilities.
It can be IR controlled, and the difficulty is to make stable normal or upside down figures.
A simple scheme is proposed, but IR handles with joystick exist.
Using the Gy521 accelerometer and gyro removes the frustration of developing pole balancing robots: Witty will never fall and you can play lots of new illuminated tricks.



The card is indeed an Arduino AVR328 compatible board with an external serial to USB module.

AVR328 pins

Pin	Port		Pin	Port	
0	PD0	Rx	11	PB3	
1	PD1	Tx	12	PB4	
2	PD2	IRmodule	13	PB5	
3	PD3		14	PC0	Pous low
4	PD4	RecG	15	PC1	Led high
5	PD5	AvG	16	PC2	Ana pin4 option
6	PD6	AvD	17	PC3	Ana pin3 option
7	PD7	RecD	18	PC4	I2C SCL pin3 1306
8	PB0	Apa Da	19	PC5	I2C SDA pin4 1306
9	PB1	Apa Ck			
10	PB2				



Software for pure Arduino fans

The table above gives the pins for the signals you wish to use.
A Led is wired on pin 15/A1, active high
A button is wired on pin 14/A0, active low. It needs a programmed pull-up resistor.
Motor are wired on pins 4,5,6,7. Use analogWrite on pins 5 and 6 and clear pins 4 and 7.
Bidirectionnal PWM can be used as explained on
https://playground.boxtec.ch/doku.php/motor/bidirectional_motorcontrol_arduino
or (same file) https://www.didel.com/robots/MotorControl.pdf
You know about PFM advantages? https://www.didel.com/PFMversusPWMforRobots.pdf

APA102 RGB strip of 8 is wired on pin 9 (ck) and pin 8 (data). Apa102 is compatible with SK9822, which is indeed the chip used on the Witty.
There are many Apa102 libs. Use e.g. the Pololu lib https://github.com/pololu/apa102-arduino

Gy521 (MMU 8050) is wired on usual I2C pins: pin18/A4 SDA pin19/A5 SCK.
The Gy521 use the Wire lib and is well documented. Find a test program on the web, e.g.
https://playground.arduino.cc/Main/MPU-6050/

Enjoy developing your own software. One function at a time is rewarding.
If you program the same as the Demo2.ino you should end-up with 20k of code.
We explain now our approach, the Demo2 is 4k bytes and we guess rather easy to read. Do not be afraid with the many .h files, it is a split of the many pages such a program requires.

Didel Witty software

All **.pdf** links easy to click on www.didel.com/Witty.html

Didel software is based on optimized simple C included files you can load as an Arduino lib, see https://git.boxtec.ch/didel/Witty
The include files are on the Arduino lib space, but since our lib are easy to be tailored to you needs, it may be good to have them attached to the main program. See www.didel.com/IncludeFiles.pdf if you are not familiar.
You played with the Demo program. It grows from 2k to 4k just by adding non interesting fancy and decorative features. If you want to understand the libs we have optimized for you, play first with the simple test programs you find on the lib.

If you want to have all these programs accessible from Arduino lib, follows the usual procedure:
Load Arduino and click on Sketch – Include a lib – Add a zip file
Add the WittyLib.zip file. Quit Arduino and reload Arduino (it needs to rebuild links).
Now under File – Examples go down to the list, click on WittyLib and you see test programs.

Modify the test programs while reading the documentation on the ".h" files.

Witty simple tests **all files from WittyTestLibs.zip**

`Stencil.ino` Change name and ready to select the include files you need.
You have to include them from the same directory

`TestWitty.ino` Test Button, Led Cligno and CliErr. Use Witty.h
When Led is on, depress button to make it last for 2 more seconds. If not, blinks and CliErr fuction, to be used when the program goes to a wrong or expected place (debug help).

`TestGetPush.ino` Use GetPush() to control the motors motors
1 push blink
2 pushes move motors
3 pushes test tv remote, quit if push
>3 Cli error: bursts with the number of pushes

`TestTelec.ino` Blink number of pulses. Use Telec.h
Push n times and check. Modify `DelSil` and `DelCnt`

`TestPfm.ino` Minimum blocking Pfm test. Use Pfm.h
DoPfm must be called every 2ms. Try 1ms, 20ms.

`TestInter2Pfm.ino` Speed change Use Pfm.h and Inter2.h
DoPfm is called by interrupt every 2ms

`TestInter2.ino` Blink by interrupt Use Inter2.h
Function DoCligno(); is called every 1ms. Its parameters are global, modified by the main program.

`TestApa.ino` Test Apa102
See where to change the Rainbow? – not trivial- Test other functions .

`TestGy521.ino` Display gyro values
Ready to do the interesting things?

Didel Witty libs functions, links and name of test programs

`Witty.h` Defines pins and basic actions (e.g on/off motors). It includes the Arduino delay(); replacement we do not use for portability to other C machines. Using delay() add 150 bytes to the program.

Witty.h `#include "Witty.h"  SetupWitty();`	**`TestWitty.ino`**
AvG; RecG; StopG;	AvG=Avance Gauche, etc (sorry, it is french)
AvD; RecD; StopD;	StopG/D block the motor but small difference with free running
LedOn; LedOff; LedToggle;	***These are macros. No action if you write AvG();***
PushOn *Boolean value*	if (PushOn) { do…}
IrmOn *Boolean value*	if (! IrmOn) { wait for IR signal }
DelMs(v16); *Max 65s*	replace delay(); you should not use
Delus(v16); *Max 65ms*	replace delayMicroseconds(); you should not use
Cligno(n8,p16); *n pulses of period p*	Blink n times, period p - do not use Cli();
CliErr(n8,p16,s16);	Blink a sequence till reset. Use when program should not go there.

Note: Start, Stop are reserved for I2C, do not use.

Test these functions with the next TestGetPush program.

GetPush.h

`GetPush.h` `#include "GetPush.h"  no Setup`	**`TestGetPush.ino`**
`nPush = GetPush();`	Blocking. Blink fast while waiting for pushes. Echo number of pushes

GetPush save time when developing an application. Many simple tests are required to understand software primitives and master hardware reactions.

Save most of your tests under a new name, you will be happy to retest them later

Apa102.h Controls the RGB strip, see https://www.didel.com/WittyApaSk.pdf for details

`Apa102.h` `#include "Apa102.h"  SetupApa102();`
`#define  Npix 8  // nombre de LEDs`
`#define  Brt 2 // max 31 brigtness`
	`TestApa102.ino`	

`ApaClear();`
`ApaRGB(r,g,b);` //values 0 to 255
`ApaLogRGB(r,g,b);` //values 0 to 16
`ApaLogRed(v);` same for `Green Blue, Yellow, Pink, Cyan, White.`
`ApaRainbow();` one solution for 8 RGBleds
	Clear the strip
	Same color on all leds
	Same color on all leds, log progression
	One parameter for 7 simple colors.
		GREEN
		YELLOW
		RED
		WHITE
		CYAN
		MAGENTA
		BLUE
		Additive Colors

Low level functions allow to set individual leds to any color. Tables can be defined and specific functions written, like the binary counter of Demo Test 5.

Telec.h Is a simple "Morse" control compatible with any TV remote

see https://www.didel.com/TelecommandeIrSimple.pdf

`Telec.h` `#include "Telec.h"  no setup`	**`TestTelec.ino`**
`nPulse = CompteIR();`	Blocking. Green Led echo the pulses received.

`DelSil and DelCnt` can be reduced for faster reaction.

Proportional control needs a handle with pots, as proposed on https://www.didel.com/Ir/RolloverCommandeArduino.pdf
SendEmir.h and GetEmir.h libs are available (3 channels). Hermes board under development.

Inter2.h Interrupt required for PFM and TerGet

See https://www.didel.com/C/Interruptions.pdf new text being prepared

`Inter2.h` `#include "Inter2.h"  SetupInter2();`	**`TestInter2.ino`**
none	The Setup starts the blink activity.

Single interrupt is easy to master. One works with interrupt flags and not with interrupt service routines (ISR), see for instance test TerGet.ino

The processor is interrupted every 60 microseconds, For most task this is too frequent and predividers call tasks at the appropriate period of time.

Pfm.h Pfm blocking test

See https://www.didel.com/prof/PwmPfm.pdf https://www.didel.com/PFMversusPWMforRobots.pdf

Pfm.h `#include "Pfm.h" SetupPfm();`	**TestPfm.ino**
`DoPfm ();` `//`no parameter, global variables PfmL,PfmR	

Main program or an interrupt routine modify PfmL and/or PfmR. The new value is taken into account at the next interrupt.

Pfm.h and **Inter2.h** Speed control

See https://www.didel.com/PFMversusPWMforRobots.pdf
https://www.didel.com/diduino/PfmPratique.pdf

Pfm.h `#include "Pfm.h" SetupPfm();`	**TestInter2Pfm.ino**
`DoPfm ();` `//`no parameter, global variables PfmL,PfmR	

Main program or an interrupt routine modify PfmL and/or PfmR. The new value is taken into account at the next interrupt.

I2cTwi.h Transfer with the Gy521

see www.didel.com/LibXI2C.pdf

`I2cTwi.h #include "I2Ctwi.h" SetupI2Ctwi();`	TestGy521.ino

`aadd = 7 bit address`
`writeByte(data);`
`data=ReadByte();`
`writeByteAt(reg,data);`
`data=ReadByteAt(reg);`
`writeWordAt(reg,data);`
`data=ReadWordAt(reg);`
	The option here is there is only one I2C channel defined in main PP. This avoid to repeat the I2C address on all transactions.

I2C block if no I2C device is attached. See www.didel.com/CheckI2C.pdf

Gy521.h Gyro/Accelero sensor

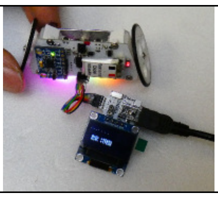
Gy521.h setup initialize the Gy521. It is not yet clear if averaging functions should be added there or as different libs.

TerSer.h replaces Serial.print() function. See https://www.didel.com/TerSer.pdf and Terser examples on https://git.boxtec.ch/didel/TerSer

Other set of useful functions are being prepared to facilitate programming Glis.h (moving averages) , Time.h (replace millis(); with directly usable time/timout counters).
Many other libs are specific to robots (Enco.h) and sensors (DistIr.h, Uson.h, DHT.h, etc).

More to help debugging

Playing with the speed of the motors is easy. Using the gyro needs a deep understanding of its behavior, that is observing the raw data produced by the Gy521, on the terminal, and on an Oled as made possible by replacing Gaia by the Pythie module. In both cases, you will appreciate the 5-line link between Witty and Gaia or Pythie.
A flexible wire can be used to link Pythie and Witty, giving enough freedom to Witty to hold it in different positions, observe variables and adjust parameters before giving Witty its full liberty (picture below).

The trick to use the Oled SSD1306 to display any useful information is to use the I2C "bitbang" library I2CBb01.h together with OledPix.h for drawing graphics and TerOled.h for displaying numbers. Scl and Sda signals are inputs only to the Oled but may disturb the downloading of programs, due to the I2C pull-ups on the 1306.	

Pythie is documented on https://www.didel.com/Pythie.pdf
Pythie replaces Gaia and takes benefit of the two lines RxD and TxD to help debugging.

jdn 190507