http://www.didel.com/ info@didel.com

www.didel.com/WittyGyro.pdf

Witty gyro and accelerometer

How to use the Gy521

1 Introduction

Gyro are used to stabilize drones, using a good modelization and tricky programming.
Many balancing robots are available on the market; they rarely satisfy their users, spending time trying to adjust parameters and put the robot back up.
The Witty is indeed more difficult to program, but it never falls down! While moving, motors fight against a linear inertia and a rotational inertia. Start slowly, it moves as a simple robot. Put full power, it rotates. Imagine all the tricks you can play with, taking care of the accelerator and gyro values and getting the desired or a surprising behavior.
We provide with the Witty a stick attached with magnets that can either convert it as a 2-wheel robot to play wit the RGB strip, or as a traditional balancing robot.

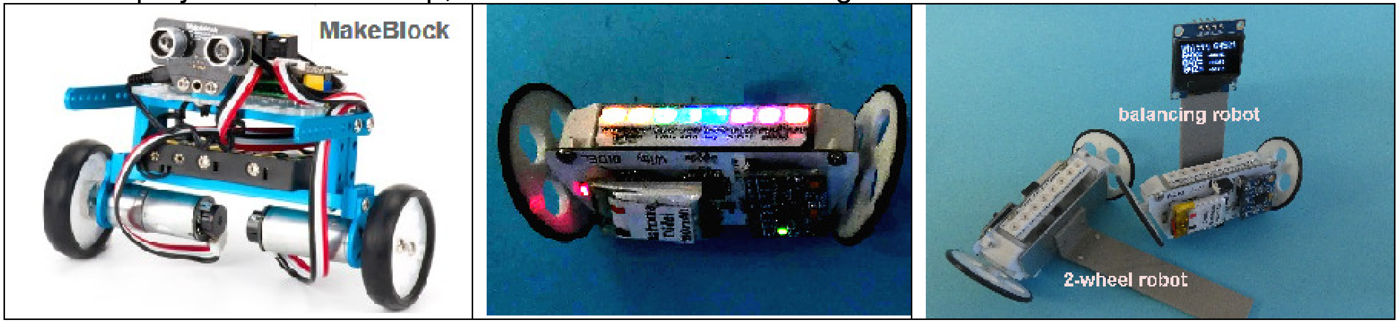


Fig 1 Traditional balancing robot and Witty options

Mastering the stability problems is delicate. One can find theory with pretty formulas, there are libraries that apply these formulas in floating point. The problem is that a calculation can only give accurate results if the data is accurate and if one can act accurately. Beside of this, the AtMega328 has only an 8-bit multiplier, no accelerator for floating point and can only do good things if programmed the adequate way.
Our project uses a Gy521 and 6mm geared motors, the goal was to build small and reactive. Light weight makes our design more difficult. Less inertia is good for the project, and inertia can be added if it helps, not removed.

2 Accelerometer and Gyro

An accelerometer measures gravity along three axes. If an axis is exactly vertical, the gravity is 1 and varies little with the angle. On both perpendicular axes, gravity is zero and varies very rapidly. With these three signals we can know the position of the sensor in space. But they are terribly noisy and must be filtered, which introduces delays.
In addition, the accelerometer measures both static and dynamic accelerations.
One need to use a low-pass filter to get the gravity and a high-pass filter to get the dynamic acceleration. Three-axis accelerometer as the Gy521/MPU-6050 provides also the gyro values on the three directions x, y, z.

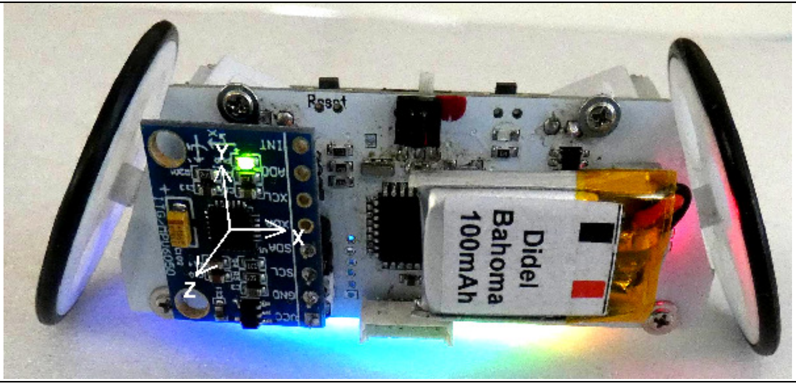


Fig 2 Gy521 sensor axes on Witty

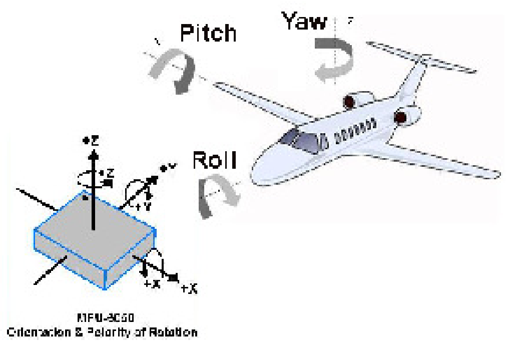


Fig 3 Traditional namings

On the Witty, some values are more interesting to know. In the position of the picture, acceleration Y is close to -1G, Z close to zero. X will stay at zero on the usually flat test surface but the roll value is interesting to know. Play with Demo program 4 and the terminal to get familiar with the accelerator values the Gy521 provides. Too many bits are calculated compared of the precision of the measure.

The gyro values are more difficult to handle. By knowing the initial position and integrating the variations, we obtain the current position. But values are very noisy and need to be filtered.

3 The MPU6050 sensor on the Gy521 module

The Gy521 module accepts a voltage of 3 to 5V, which is necessary to develop at 5V and then navigate to 3.7V having possibly retouched the parameters.
The doc of the MPU6050 is scary with 120 control registers at first glance. In fact all these registers have a correct default value for our use, except one, necessary to wake up the circuit. Two other registers will be commented on later.
The module is I2C, we must add the pull-up resistor (4k7). We can ignore the 4 additional signals XDA, XCL, ADC and INT.

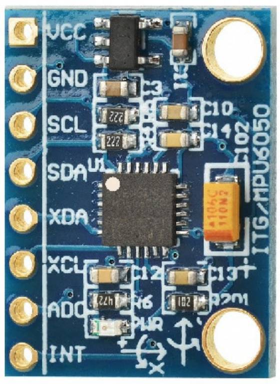
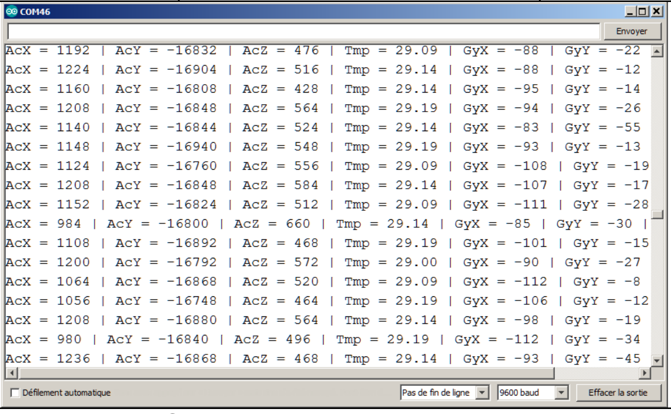


Fig 4 Gy521 module based on the MMU6550

4 Reading the MPU6050 sensor

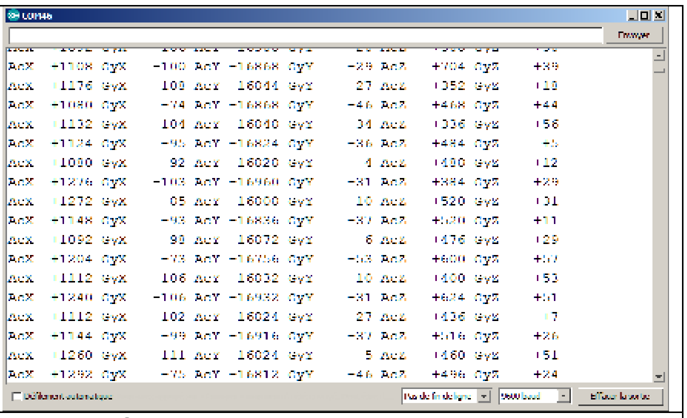
The set-up of the MPU6050 is simple; Its I2C address is 0x68 (7-bit address) and for standard operations, there is only register 0x6B to clear.

The data of the sensors, plus the temperature that can be ignored, are prepared in 16-bit registers. The internal 6050 controller update these registers every 4ms (260Hz) or less depending register xx.
Arduino document under https://playground.arduino.cc/Main/MPU-6050 the program that displays the values of the registers on the Serial terminal (5698 octets, 490variables).
You can find on Witty lib the equivalent program (TestGy521.ino) using TerSer.h and the I2cTwi.h lib (1616 octets, 67 variables). Demo2.ino test 4 shows the same.



```
AcX = 1192 | AcY = -16832 | AcZ = 476 | Tmp = 29.09 | GyX = -88 | GyY = -22
AcX = 1224 | AcY = -16904 | AcZ = 516 | Tmp = 29.14 | GyX = -88 | GyY = -12
AcX = 1160 | AcY = -16808 | AcZ = 428 | Tmp = 29.14 | GyX = -95 | GyY = -14
AcX = 1208 | AcY = -16848 | AcZ = 564 | Tmp = 29.19 | GyX = -94 | GyY = -26
AcX = 1140 | AcY = -16844 | AcZ = 524 | Tmp = 29.14 | GyX = -83 | GyY = -55
AcX = 1148 | AcY = -16940 | AcZ = 548 | Tmp = 29.19 | GyX = -93 | GyY = -13
AcX = 1124 | AcY = -16760 | AcZ = 556 | Tmp = 29.09 | GyX = -108 | GyY = -19
AcX = 1208 | AcY = -16848 | AcZ = 584 | Tmp = 29.14 | GyX = -107 | GyY = -17
AcX = 1152 | AcY = -16824 | AcZ = 512 | Tmp = 29.09 | GyX = -111 | GyY = -28
AcX = 984 | AcY = -16800 | AcZ = 660 | Tmp = 29.14 | GyX = -85 | GyY = -30 |
AcX = 1108 | AcY = -16892 | AcZ = 468 | Tmp = 29.19 | GyX = -101 | GyY = -15
AcX = 1200 | AcY = -16792 | AcZ = 572 | Tmp = 29.00 | GyX = -90 | GyY = -27
AcX = 1064 | AcY = -16868 | AcZ = 520 | Tmp = 29.09 | GyX = -112 | GyY = -8
AcX = 1056 | AcY = -16748 | AcZ = 464 | Tmp = 29.19 | GyX = -106 | GyY = -12
AcX = 1208 | AcY = -16880 | AcZ = 564 | Tmp = 29.14 | GyX = -98 | GyY = -19
AcX = 980 | AcY = -16840 | AcZ = 496 | Tmp = 29.19 | GyX = -112 | GyY = -34
AcX = 1236 | AcY = -16868 | AcZ = 468 | Tmp = 29.14 | GyX = -93 | GyY = -45
```

Fig 5 Arduino Serial – data not aligned



```
AcX  +1108 GyX   -100 AcY -16868 GyY    -29 AcZ   +704 GyZ    +39
AcX  +1176 GyX    100 AcY  16044 GyY     27 AcZ   +352 GyZ    +18
AcX  +1080 GyX    -74 AcY -16868 GyY    -46 AcZ   +468 GyZ    +44
AcX  +1132 GyX    104 AcY  16040 GyY     34 AcZ   +336 GyZ    +56
AcX  +1124 GyX    -95 AcY -16824 GyY    -36 AcZ   +484 GyZ     +5
AcX  +1080 GyX     92 AcY  16020 GyY      4 AcZ   +400 GyZ    +12
AcX  +1276 GyX   -103 AcY -16960 GyY    -31 AcZ   +384 GyZ    +29
AcX  +1272 GyX     85 AcY  16000 GyY     10 AcZ   +520 GyZ    +31
AcX  +1148 GyX    -93 AcY -16836 GyY    -37 AcZ   +520 GyZ    +11
AcX  +1092 GyX     98 AcY  16072 GyY      6 AcZ   +476 GyZ    +29
AcX  +1204 GyX    -73 AcY -16756 GyY    -53 AcZ   +600 GyZ    +57
AcX  +1112 GyX    106 AcY  16032 GyY     10 AcZ   +400 GyZ    +53
AcX  +1240 GyX   -106 AcY -16932 GyY    -31 AcZ   +624 GyZ    +51
AcX  +1112 GyX    102 AcY  16024 GyY     27 AcZ   +436 GyZ     +7
AcX  +1144 GyX    -99 AcY -16916 GyY    -37 AcZ   +516 GyZ    +26
AcX  +1260 GyX    111 AcY  16024 GyY      5 AcZ   +460 GyZ    +51
AcX  +1292 GyX    -75 AcY -16812 GyY    -46 AcZ   +496 GyZ    +24
```

Fig 6 TerSer - data aligned

5 Witty motors

As detailed on WittySpecs.pdf, the Bo30 motor has a gear factor of 1:96 and a wheel of 32mm. Speed and acceleration have been evaluated using a 16-slot encoder on a similar motor (no load). The oscilloscope shows full speed is reached after less than 60 degrees of the shaft rotation, and speed is 200ms/turn, that is 5t/s, or 50 cm/s.

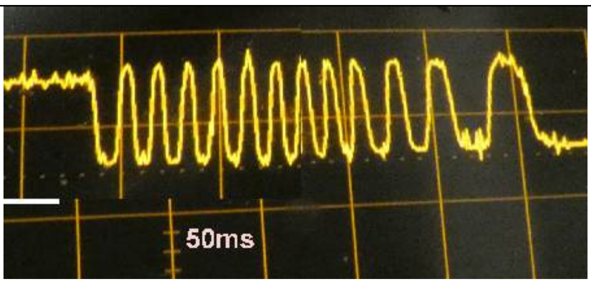


Fig 7 Bo30 start/stop with 16-slot encoder

Whitty specs explain how to control the motor on/off, PWM and PFM.

6 Control theory

Control theory uses beautiful mathematics and professional tools like MathLab and LabView running on powerful processors. Reading sensors, removing noise, applying control models, setting the motor parameters imply a set of tasks that must be the smallest amount of time, to be reactive. This mean every task must be under control for its execution time. Use floating point, ok, but what is the reaction time? Has my robot enough inertia to accept such a low response time?
We have to program the Witty with that constant idea: keep the soft compact and fast.
.

7 Control principle

The general control problem is to to have a motor that moves a mass and need to reach a position as quickly as possible and be stable there after a minimum amout of overshoot and oscillations.
The drawing makes clear what is an excellent evolution and apply to a servo as well as for a balancing robot.

Let us suppose, as for traditional balancing robot, we want to keep the Witty vertical. If it leans forward, it must move forward faster. If the speed is proportional to the error (the angle) one talk about a setting P, P for proportional.
When stable, an effort, the friction, the fact that it was not balanced initially, can prevent reaching the desired position; it's the static error. The integral of the error would increase if there is not a correction from the sum of the differences between the desired position and the successive positions. It is the setting I, I for integration.

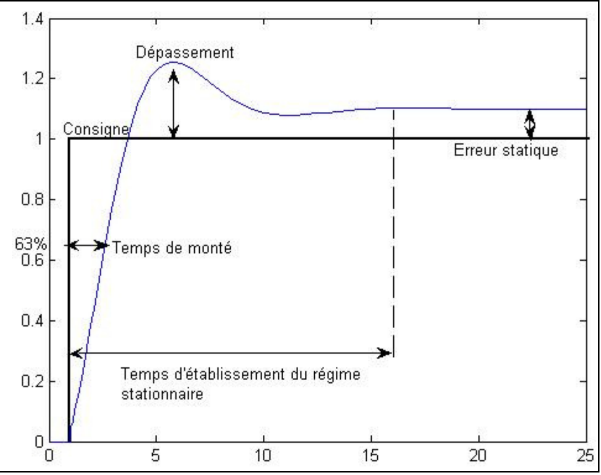
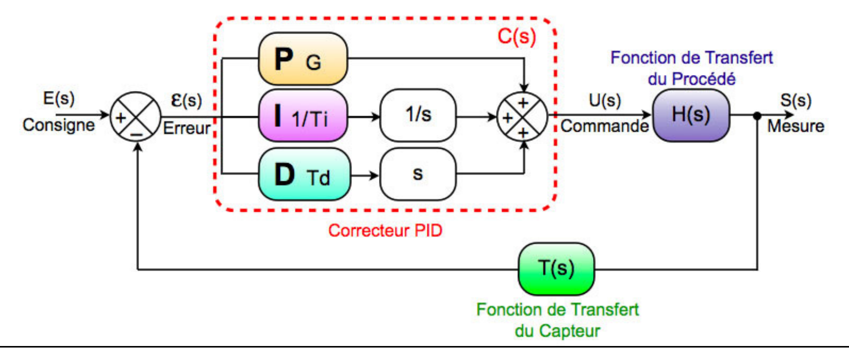


Fig 8 Typical regulation curve.

One can still worry about oscillations, and take into account the variation of the gap. It is the component D, D for derived.
We then have a complete PID setting. In some case, the D is dropped because the sensor fluctuates too much and too quickly, in some other cases the I is not recommended. Anyway, PID proportional control is based on multiplication coefficient, as shown on this diagram:



e(s)	error value
U(s) =	motor control
Kp∗e(s) +	prop coefficient
Ki∗int(e(s)) +	integral coefficient
Kd∗diff(e(s))	differential coefficient

Fig 9 The basic control algorithm

The value of the coefficient depends on the inertia and the motor characteristics, that is available that connect be evaluated without adequate equipments. If one have to guess and correct these parameters, why to use precise floating point multiplications that slows down the reaction time? Multiplications by n/16, n/64 are much faster and n/64 is almost 1%, when all the rest will be 10%.

8 Sensor's noise

To reduce the noise of the sensors, it is necessary to average and there are three simple ways to average, plus very complex techniques. The simple average sums up e.g. 4 measurements and gives a result 4 times less often, which is obviously not favorable. The sliding average measure the average of the previous 4 measures.
Weighted average give a weight to the stored measures, so it is possible to react faster or slower to the last measure, and get something like a low pass or high pass filter. See the web for details and our document www.didel.com/Averaging.pdf).

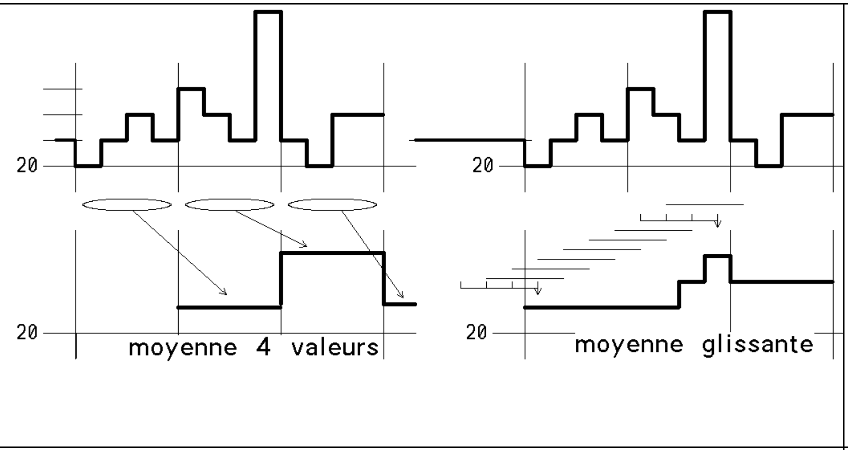


Fig 10 Average and sliding average on 4 values

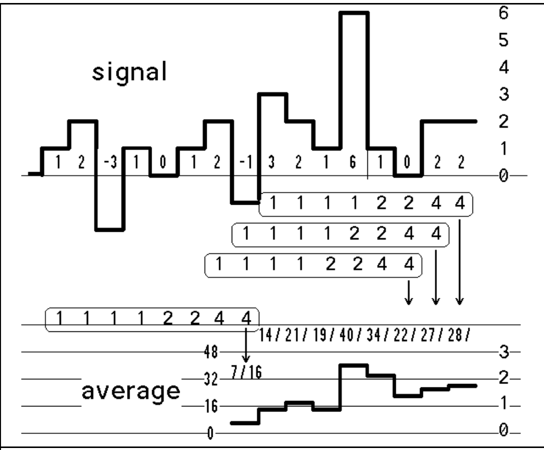


Fig 11 Weighted average on 8 values

9 Filtering
Averaging and filtering are not so different; filtering can be seen as a refinements of weighted average. As shown on the weighted average example above, a greater importance is given to the lastly received values. This correspond to a high-pass filter, fast changing values have a greater influence.

10 Derivate

To derivate, one need to calculate the difference between two consecutive values. The current value is saved to be used for calculating the next difference. At initialisation, the saved value is the current value.

Differentiate

```
//Variables
int  v8, prevV8;
intdiff;
//Each time vs is read:
diff =  v8 - prevV8;
prevV8 = v8;
```

11 Integrate

To integrate, one just need to add the difference between the desired position (Goal) and the current position.
Overflow are difficult to handle; it may be good to saturate at a value depending on the application..

Integrate
integr += (v8-vGoa)l;

12 Sample time
As shown in figure 11, the calculation of the average is done regularly. Sensors have a response time., e.g measuring the Acz and GyY parmeters of the Gy521 takes 700 µs. Let us take 2ms to be compatible with PFM cycle. Sample times of 20ms are frequently mentioned for 200 grams balancing robots. The witty is lighter and need faster reactions.

13 Preparing the variables

As shown in figure 2, we need AcZ and optionally GyX as a derivate of AcZ.
We will have to integrate the error of AcZ relative to the "vertical position" at start-up.
Hence we will calculate the sliding average of AcZ and GyX, named mAcZ and mGyX.
We need to have an initial value of AcZ and GyX, named iniAcZ and iniGyX
We need to integrate, accumulate AcZ error, named acumAcZ
We may test the used of the differential of AcZ in place of GyY, variable will be dAcZ.
Having these variables calculated, analysed separately to see the range and errors, it will be possible to test PID formulas, the difficulty being to estimate 2 or more parameter.

14 Averaging AcZ and GyX

Let us use Glis.h library, doc on www.didel.com/Average.pdf
We need to initialize the sliding average table, remember the initial position and play moving the Witty to see how the measured values change.

(This program is part of the test programs on Witty lib)

```
//TestGlis.ino  Display values and average
//use TerSer
#define aadd 0x68*2 // 8-bit addr

//variables AcX ..defined in Gy521.h
volatile int16_t iniAcZ,mAcZ;
volatile int16_t iniGyX,mGyX;
volatile int16_t  acumAcZ;

#include "Witty.h"
#include "TerSer.h"
#include "I2Ctwi.h"
#include "Gy521.h"
#include "Glis.h"
 #define LGli 8
void setup() {
  SetupWitty();
  SetupI2Ctwi();
  SetupGy521();
  SetupTerSer();
```

```
   DelMs(1000); // Put in place/stabilization time
   AcZ=ReadWordAt(0x3F);
   IniGlis1(AcZ); iniAcZ=AcZ;
   GyX=ReadWordAt(0x43);
   IniGlis2(GyX); iniGyX=GyX;
}

void loop() {
    Text("Gy521 initial values"); CR();
    Text("iniAcZ "); Dec16s(iniAcZ);
    Text("       iniGyX "); Dec16s(iniGyX); CR();
       while(1){
          AcZ=ReadWordAt(0x3F);    GyX=ReadWordAt(0x43);
          Text("   AcZ "); Dec16s(AcZ);
             Dec8s((AcZ-iniAcZ)/16); Text("%%");
          Text("   GyX "); Dec16s(GyX);
             Dec8s((GyX-iniGyX)/16); Text("%%"); CR();
          DelMs (500);
       }  // end while
}  // end loop
```

As shown on the terminal screen, the program display the initial position and then repeat the measures. The difference with initial position is shown in ‰, so on can see it is infrequent to reach 1%, that correspond to 0.6 degrees. At the extremity of a 10cm pole, this means 1mm.

Always think relative and not absolute!

Did you notice the "defilement automatique" case on the terminal screen.

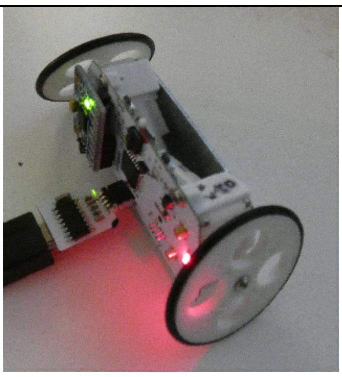
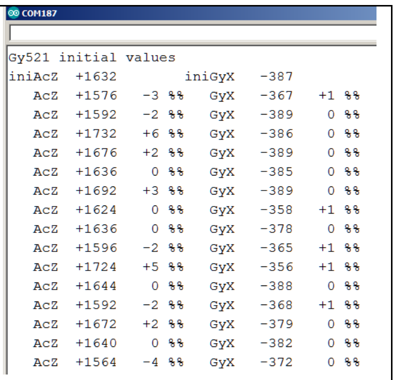


Fig 12 Downloading to Witty



```
Gy521 initial values
iniAcZ  +1632        iniGyX  -387
   AcZ  +1576  -3 %%   GyX  -367  +1 %%
   AcZ  +1592  -2 %%   GyX  -389   0 %%
   AcZ  +1732  +6 %%   GyX  -386   0 %%
   AcZ  +1676  +2 %%   GyX  -389   0 %%
   AcZ  +1636   0 %%   GyX  -385   0 %%
   AcZ  +1692  +3 %%   GyX  -389   0 %%
   AcZ  +1624   0 %%   GyX  -358  +1 %%
   AcZ  +1636   0 %%   GyX  -378   0 %%
   AcZ  +1596  -2 %%   GyX  -365  +1 %%
   AcZ  +1724  +5 %%   GyX  -356  +1 %%
   AcZ  +1644   0 %%   GyX  -388   0 %%
   AcZ  +1592  -2 %%   GyX  -368  +1 %%
   AcZ  +1672  +2 %%   GyX  -379   0 %%
   AcZ  +1640   0 %%   GyX  -382   0 %%
   AcZ  +1564  -4 %%   GyX  -372   0 %%
```

It is interesting to see the effect of the size of the average, statically and dynamically.

15 Integrate AcZ

We may need to integrate the AcZ value. Variable acumAcZ will add the differences we have already calculated. The work is to reorganize the display to show that value and see how it change when the Witty is slightly pushed.

16 Derivate AcZ

We may need to derivate AcZ and it will be interesting to compare with GyX values. The scale however has no reason to be the same, that is one idea is to compute the ratio and see how it change. Sampling rate may play an important role, interesting to test.
Note: we assume your interest is not to load programs, but write them and get a complete understanding of the interaction between software and hardware. You should be able to continue by yourself, if not, let us know what are the difficulties you are facing with.

17 Speed control

For a good control of the motor, one need in addition to the gyro data, a high resolution encoder, since stability corrections need very precise movements of the wheels; but the speed is the derivate of the number of pulses per sample time; we need very many pulses per turn and only very expensive and voluminous industrial encoders could be used.
We have to assume the wheels will move at a speed proportional to the motor power. This will work better if the inertia at the level of the motors is low. If the inertia far from the motors is high, it will facilitate the parameter's adjustment.
Proportional control of the motor can be achieved with three algorithms
- PWM with the drawback that the minum pwm value may be 10% or more
- PFM with the drawback the the next pulse may take some time to arrive, if low PFM value

- Not yet tested solution with a motor pulse length duration that starts immediately when the PID control value is calculated.

The programs to experiment with motor speed are easy to write and test programs that creates small oscillations will surely be helpful later. An oscilloscope is a powerful tool; Didel works on debugging helps (Pythie, Janus). Real time is of course the essential condition.

18 P algorithm
Proportional control is now easy to test. Pwm or pfm value is proportional to the position error, as calculated in section 14. Since the dynamic cannot be modellized, successive approximation will come close to the balancing objective.

19 PI, PD, PID algorithm
Adding the integral error should increase the stability. We believe that studying "naively" the effect of integral and derivative will help you to understand the literature about PID.
Here are some links, usually assuming that measures are perfect and floating point calculations will do the job in time.

https://positron-libre.blog/robots/pendule-inverse-vertibot.php
https://www.pc-control.co.uk/pid_control.htm
https://theautomization.com/pid-control-basics-in-detail-part-1/

20 Conclusion

Didel is concerned about design of educational material. The objective has never been to sell a Witty or Pegase with a balancing software you can just download and see it works. Learn how to do it is more interesting. It is complex and we may add some steps in the future, hoping customers will help providing the base material.

Appendix
Gy521 Optimization
The register 25 = 0x19 Sample Rate Divider has a value linked to that of the register 26 = 0x1A Configuration. The default value is 0 and known applications do not attempt to modify it. A specialist advises
R25 value 22
R26 value 4 (going to 6 should slow down the reaction since the filtering delay changes to 19ms instead of 20ms).

Jdn 190514