///Notre robot est surtout un projet personnel éducatif réalisé dans le but d'avoir une expérience pratique dans l'utilisation et la manipulation des filtres de Kalman et des paramètres d'un régulateur PID.

//// <http://blog.positron-libre.com/robots/pendule-inverse-vertibot.php>

**////Basic control theory**

In short I am using cascaded PID controllers. The outer loop is regulating the speed of the robot which is measured with encoders on the motors. The internal loop controls the robot's angle, which is measured with an IMU. The outer loop has slow dynamics and the internal loop is much faster. Theoretically it should be possible to program a self balancing robot using only the inner loop but you get very easily a behaviour of the robot inching forward or backward across the floor because of the missing feedback from the wheel speeds.

To gain additional performance gain scheduling is used on the inner loop, which means that the regulator parameters will change depending on the robot's angle. A small angle near the equilibrium means conservative regulator parameters where only small adjustments are made to keep the robot standing. If the angle of the robot is close to falling, the aggressive paramter settings are activated in order to do everything to regain stability.

///////////The most important element of any robot is the controller. Especially for a self-balancing robot, the control program is vital as it interprets the sensor data and decides how much the motors need to be moved in order for the robot to remain upright. The most common controller used for stabilisation systems is the PID controlle

/////Now-days gyroscopes are extremely small and very cheap to buy, so they are ideal for amateur electronics projects. Unfortunately these gyroscopes (both the cheap and the not-so-cheap versions) also come with their own problems. They are good for short-term and quick movements, but tend to drift over time as the error accumulates. They also record a lot of jitter and noise, which needs to be filtered by the micro-controller before the data can be used.

To reduce this drifting effect of the gyroscope, it is possible to combine the sensor data with that from an accelerometer. The accelerometer is good at sensing slower and more prolonged movements, rather than the fast motion. Therefore if we take the best of both worlds and fuse the data together, we will be left with an extremely accurate picture of the motion of the robot.

As a result I decided to use a combined accelerometer & gyroscope breakout module (the MPU-6050), which is slightly more expensive than a simple gyro, but should lead to a superior stabilisation performance. Note: the MPU-6050 comes with a library which does all of the calculations and filtrations for you, so that definitely is a plus!

/////////**So what’s happening in that sketch? In the function demoOne() we turn the motors on and run them at a PWM value of 200. This is not a speed value, instead power is applied for 200/255 of an amount of time at once**.

///