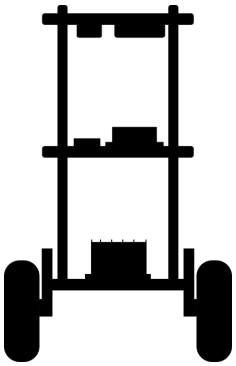
POLITECNICO DI MILANO

Department of Electronics, Information and Bioengineering

Polo Territoriale di Como

Ercolino: Self-Balancing Robot

Model Identification and Adaptive Systems Project



Authors:
Alessandro Castiglioni
Davide Molinelli

Supervisor:
Luigi Piroddi

March 24, 2018

Contents

1 Abstract **1**

2 Problem Description **1**

3 Robot Construction **2**

4 Accelerometer and Gyroscope **3**
4.1 Calibration procedure 3
4.2 Accelerometer angle computation 3

5 Kalman Filter **5**
5.1 The system state 5
5.2 The system output 7
5.3 The system model 9
5.4 The prediction phase 9
5.5 The update phase 10

6 Motor Control **12**

7 Conclusion **13**

1 Abstract

Ercolino is an Academic Project commissioned in the context of the Model Identification and Adaptive Systems course. The project aims to realize a two-wheeled structure, programmed with a micro-controller, able to balance itself and maintain the equilibrium. The project makes use of a Kalman filter implemented in order to extract the exact angle of inclination of the robot from an accelerometer and a gyroscope, two sensors affected by noise and drift problems. The obtained information is finally used to drive the motors and correct the incline of the robot.

2 Problem Description

Equipped with an accelerometer and a gyroscope, the robot is able to measure the acceleration and the angular velocity that it assumes in the three dimensions of the space, moment by moment. Theoretically, it would be possible to use the data registered by either of the two components (the accelerometer or the gyroscope) to determine the angle of the robot and to apply a correction in order to keep it in balance. Nevertheless, both the accelerometer and the gyroscope are subject to measurement errors:

- The mechanical components of the accelerometer are sensible to vibration, while the electronics of the component is affected by problems related to noise and quantization errors. This determines that an aleatory and unpredictable contribution is added to the correct measured value.
- The gyroscope is characterized by a drift issue, i.e. a deviation in the measured angle over time. In other words, since the computation of the current degree depends on the measurement of the angle at the previous instant, a small and apparently not relevant error is added at any new measurement. The result is that in the short term the estimate of the inclination can be considered as approximately correct (the sum of the errors does not affect significantly the calculation of the new angle), however in the long run the contribution of the error increases, resulting in completely inaccurate estimates.

In the light of the foregoing, it is necessary to rely on the data collected by both the two components and try to correct the problems associated with both instruments through filtering algorithms such as the complementary filtering[1] or the Kalman filter. Consequently, the filtered and combined data would provide the necessary information to the micro-controller, in order to properly act on the motors driving the wheels and keep in balance the robot.

[1] According to the scope of the project we decided to implement the Kalman filter which represents a course topic. However, it is interesting to know how the complementary filter works. It basically consists of a digital low-pass filter applied to the accelerometer (in order to cut the high-frequencies that corresponds to its unreliability in the short term) and a digital high-pass filter applied to the gyroscope (in order to cut the low-frequencies that corresponds to its inaccuracy in the long run). In this way, the filter relies on the gyroscope measurements in the short term and on the accelerometer measurements in the long term. Despite its simplicity, the application of the complementary filter is limited to simple applications and requires a not negligible effort in the tuning of the parameters. On the contrary, the Kalman filter presents a wide field of applicability and it is more accurate.

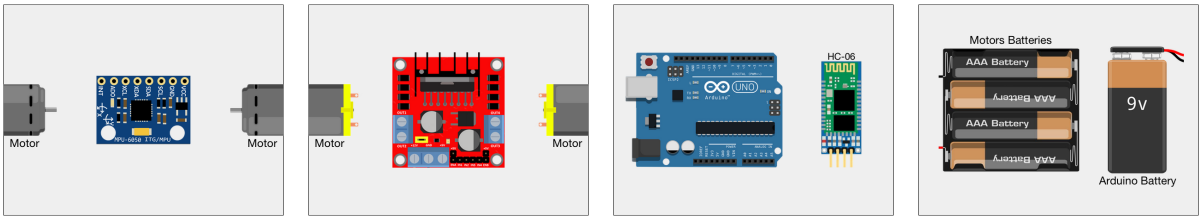
3 Robot Construction

The following is the list of the main electronic components used for the whole system:

- Arduino UNO Rev.3
- MPU-6050, a MEMS device which combines a 3-axis gyroscope and a 3-axis accelerometer on the same die.
- L298N, a Dual Full-Bridge Motor Driver used to correctly operate the motors. It accepts standard TTL logic to pick the direction of rotation and PWM signals for speed selection.
- HC-06 Bluetooth Module, which forwards the Arduino's serial port (UART) over a Bluetooth channel. This component allows us to wiressly transmit real time data from the robot to a Personal Computer and, there, visualize them over a graph or store them for off line analysis.
- Two sets of DC 3V-6V Gear Motors with robot wheels.
- 9V battery for Arduino power supply / 4 x 1.5V AA batteries for motors.

For the construction of Ercolino we've chosen a modular and extremely adjustable approach. The main structure is composed by three thin wooden boards fixed on four vertical threaded bars. Each board represents one height level and the distance between "floors" can be adjusted by acting on some nuts.

The physical arrangement of components for each level is shown in Figure 1.



(a) MPU6050 (b) L298N Motor Driver (c) Arduino and HC-06 (d) Battery Packs

Figure 1: Components displacement: (a) shows the first level, bottom view; (b) shows the first level, top view; (c) shows the second level, top view; (d) shows the third level, bottom view.

Notice that there is a reason why the heavy battery packs are placed on the top. The self-balancing robot is essentially an inverted pendulum. It can be balanced better if the center of mass is higher relative to its pivot, i.e. the wheel axle. A higher center of mass means a higher mass moment of inertia, which corresponds to lower angular acceleration (slower fall). You can think of this behavior imagining that an higher mass on the top acts like a fictitious pivot that more efficiently converts engines work in the rotation of the entire robot, rather than in a transversal movement. This applies in particular when the system is not so far from the vertical equilibrium.

4 Accelerometer and Gyroscope

4.1 Calibration procedure

MPU-6050 is an extremely cheap sensor, belonging to the family of sensors suited for detection, surely not for precise measurement. A calibration phase is necessary to remove the offset (also called bias) for each of the 6-axis.
We opted for a static calibration, which is performed at each startup with the robot laying in a fixed position; after a short settling time required by the sensor, 512 consecutive readings for all the axis are averaged together and offsets are computed.

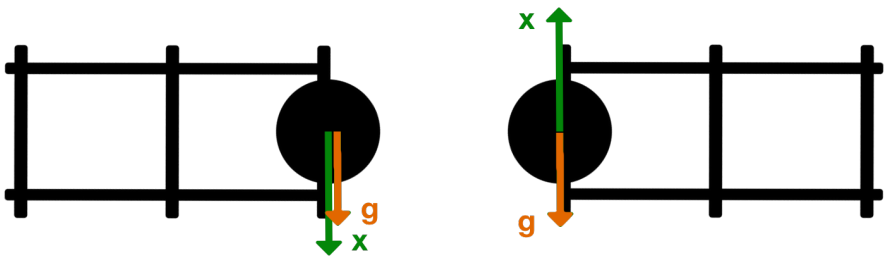
- The computation for gyroscope is straightforward: at rest condition all the readings from the gyroscope should give 0 deg/s. A deviation from this value directly represents the offset of the sensor.

- As regards the accelerometer, more about the robot position must be known. The system supports two possible starting positions, shown in Figure 2. In both configurations we expect zero acceleration on y and z axis, and the entire gravity acceleration, g, laying along the x-axis. When the x-axis is pointing downwards (figure 2(a)), g has a *negative* value, and the offset can be computed in this way:

  $$offset_{ax} = mean_{ax} + 9.81\text{m/s}^2$$

  If x points upwards (figure 2(b)), g has a *positive* value, and the formula used to compute the offset in the x measurement of the acceleration becomes:

  $$offset_{ax} = mean_{ax} - 9.81\text{m/s}^2$$



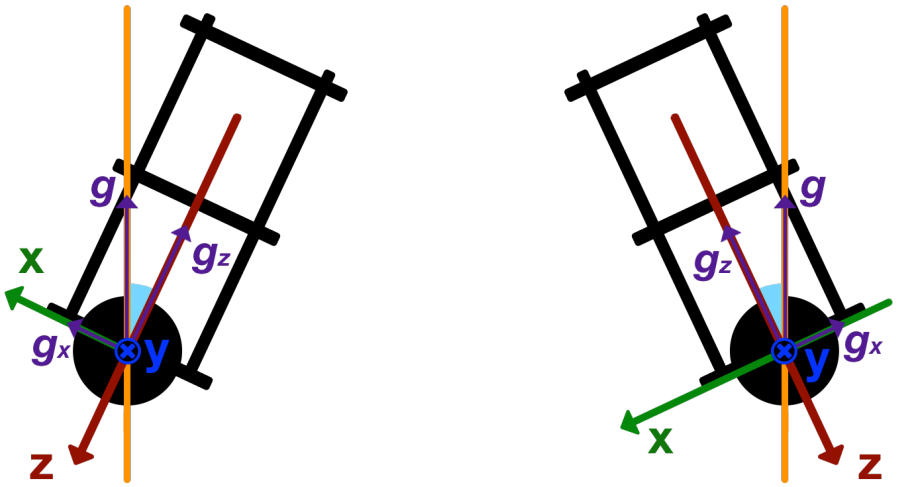
(a) x-axis pointing downwards (b) x-axis pointing upwards

Figure 2: Starting positions required for calibration

Calibration happens every time the robot is started, to guarantee the best possible performances of the sensor; in this way it's possible to adapt to different conditions in the operating temperature and to modifications in the robot structure. The entire procedure requires only a few seconds.

4.2 Accelerometer angle computation

In order to get the angle measurement from the accelerometer we need to preform some trigonometrical calculations. Before that, it's necessary to clarify the reference system associated to the robot, which is imposed by the collocation of the MPU-6050 on it.



(a) (b)

Figure 3: Tilting positions with respect to the vertical line

Remember that when the direction of the gravity acceleration is opposite to the direction of an axis, the value registered by the sensor for that axis is positive. Conversely, if the gravity acceleration and the axis face on the same direction, that value is negative.

Figure 3(a) shows the reference system of the robot when it is tilted with the x-axis up, whereas figure 3(b) shows the situation when the robot is falling in the other direction. As you can notice, in the first case the g_x component is positive, while in the second case the x-effect of gravity acceleration is negative; z-acceleration is always negative.

In this context, the angle with respect to the vertical axis can be computed applying the following formula:

$$\alpha = \arctan\left(\frac{g_x}{-g_z}\right)$$

In the setting represented in figure 3(a), since we pick the negative value of g_z, both the components are positive and the resulting angle is also positive; this is in accordance to the sign of the angular velocity obtained by the gyroscope in the event of a clockwise rotation around the y-axis (pitch). In the situation depicted in figure 3(b), instead, the component along the x-axis becomes negative, so that the angle measured through the above formula is negative. Again, this is in accordance with rates from the gyroscope.

The method shown so far works well only in the two analyzed quadrants but produces wrong results in other operational areas. Indeed, the *arctan* function can produce results only in the range $(-\frac{\pi}{2}, +\frac{\pi}{2})$ and this could seem enough for the purpose of our system. However, during the calibration phase we actually require the robot to stay fixed on its side, causing troubles in the computation of the accelerometer angle. When the incline angle is equal to ± 90 degrees, the z-acceleration is 0 and the ratio between the components has no meaning (division by zero).

In order to overcome this issue, we changed the above formulation by replacing the simple *arctan* function with the *arctan2* function:

$$\alpha = \arctan2\left(g_x, -g_z\right)$$

Arctan2 function is defined as:

$$\text{arctan2}(y,x): \begin{cases} \arctan\left(\frac{y}{x}\right) & \text{if } x > 0 \\ \arctan\left(\frac{y}{x}\right) + \pi & \text{if } x < 0 \text{ and } y \geq 0 \\ \arctan\left(\frac{y}{x}\right) - \pi & \text{if } x < 0 \text{ and } y < 0 \\ +\frac{\pi}{2} & \text{if } x = 0 \text{ and } y > 0 \\ -\frac{\pi}{2} & \text{if } x = 0 \text{ and } y < 0 \\ \text{not defined} & \text{if } x = 0 \text{ and } y = 0 \end{cases}$$

and it's not only able to distinguish between diametrically opposite angles, thus expanding the co-domain to $(-\pi, +\pi)$, but it's also defined when $g_z = 0$.

5 Kalman Filter

The Kalman filter operates by producing a statistically optimal estimate of the system state based upon the knowledge of other measurable and uncertain variables. In this project, the variable to be estimated is the angle of inclination of the robot with respect to the vertical axis, fixed by convention at 0 degrees. More in detail, the problem we are facing is a *generalized filtering problem* where the variable to be predicted depends on the measured one (of which only noisy measurements are available), but does not coincides with it.

5.1 The system state

The state of the system (call it $x(t)$) at time t is represented by the following equation:

$$x(t) = F(t)x(t-1) + G(t)u(t) + v_1(t)$$

where:

- $x(t)$ is represented by the following vector:

$$x(t) = \begin{bmatrix} \theta(t) \\ \dot{\theta}(t)_b \end{bmatrix}$$

  Note that the output of the filter is therefore represented by a couple of variables, where the first element $\theta(t)$ refers to the angle of inclination of the robot, i.e. the goal of our estimation, while the second element $\dot{\theta}_b$ accounts for the gyro bias, i.e. how much the gyro has drifted (it is an angular speed).

- $F(t)$ is a 2x2 dynamic matrix which represents the state transition model applied to the previous state $x(t-1)$. It is defined as:

$$F = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix}$$

  where Δt corresponds to the time interval elapsed between the previous step and the current one (reason why F is not a static matrix).

- $u(t)$ represents the control input of the system. In our case it is constituted by the measurement of the gyroscope, registered in *degrees/sec* and represented by the variable $\dot{\theta}$. Consequently, we can rewrite the state equation in the following way:

$$x(t) = Fx(t-1) + G\dot{\theta}(t) + v_1(t)$$

- $G(t)$ is a 2x1 dynamic matrix which represents the control input model applied to the measurement of the gyroscope, $\dot{\theta}$. It is defined as:

$$G = \begin{bmatrix} \Delta t \\ 0 \end{bmatrix}$$

  and it's used to compute the angle θ by multiplying the rotation rate $\dot{\theta}$ by the time interval elapsed from the last iteration, Δt. The second term is zero because it's not possible to calculate the bias directly based on the rate.

- $v_1(t)$ accounts for the *process noise*. It is a vector composed by two random variables with Gaussian distribution of zero mean and covariance matrix Q:

$$v_1(t) = \mathcal{N}(0, Q)$$

  More in detail, Q is a 2x2 dynamic matrix which presents the following structure:

$$Q = \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}} \end{bmatrix} \Delta t$$

  As you can note, Q depends to the current instant t, reason why it is multiplied by the delta time (this makes sense as the process noise will be larger as longer the time is since the last update of the state; for instance the gyro could have drifted). Its contribution influences the state estimate of the angle and the bias. Since we consider them independent from each other, Q matrix is actually just equal to the variance of the estimates of the accelerometer and bias (the values outside the main diagonal are equal to zero).
  It is important to dwell on the settings of the values of Q: if we fix the variances Q_θ and $Q_{\dot{\theta}}$ to large values, we introduce more noise on the state equation and the estimation results become subject to greater variability. It follows that the system does not trust the prediction of the angle and the bias: its behavior looks more responsive and aggressive, as you can see in in Figure 4, where the blue line represents the angle proceeds from accelerometer measurement while the red line corresponds to the angle estimated with the Kalman Filter.[2] The accelerometer is strongly affected by the nervousness of the system and oscillates among the fictitious values -50 and +50 degrees (it is influenced by the effect of the angular acceleration impressed to the robot by the motors that try to correct its vertical position). The angle estimated with the Kalman Filter changes repeatedly and too quickly around the 0 value, representing the vertical axis: the robot fails to correctly control the motors and soon it falls to the ground.

[2]For the test, we set the value of $Q_\theta = 1.001$ and the value of $Q_{\dot{\theta}} = 1.001$.

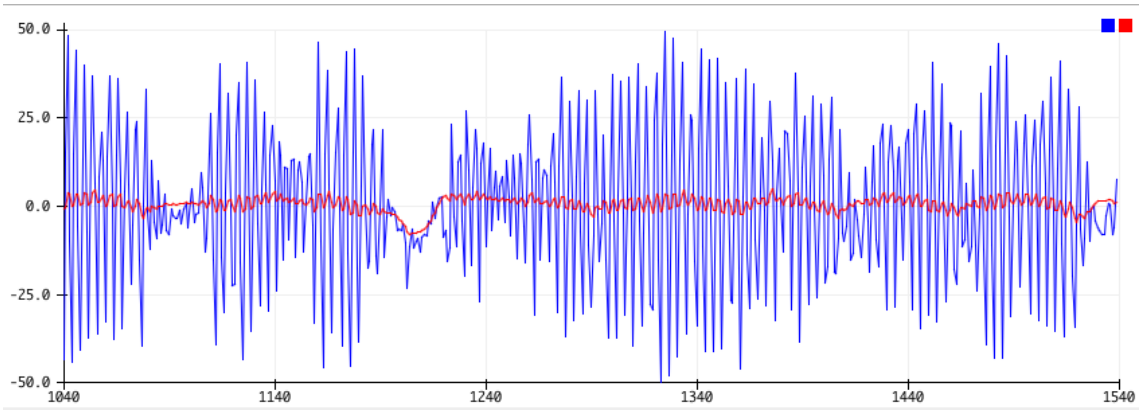


Figure 4: Setting the variance of the *process noise* too high the response of the system is very responsive and aggressive. The robot fails to correctly control the motors and soon it falls to the ground.

On the contrary, low values of variance Q_θ and $Q_{\dot{\theta}}$ reduce the variability on the estimation of the angle and the bias: we trust more the predicted state variables and, therefore, the behavior is more fluid and relaxed, as you can see in Figure 5.[3]

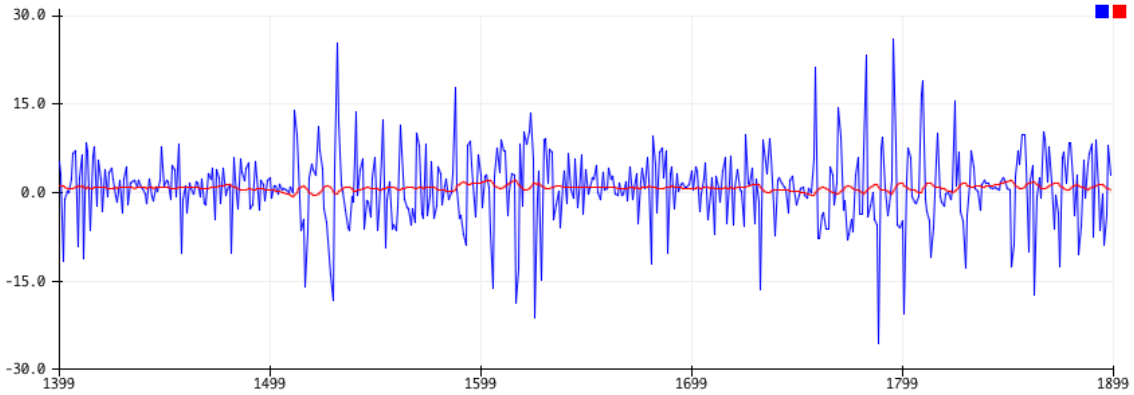


Figure 5: Setting the variance of the *process noise* to low values, the response of the system is fluid and relaxed.

The accelerometer measurement is less affected by the nervousness of the system and oscillates among a smaller range of values (the motors are subjected to fewer sudden changes in the directions of action, so the accelerometer is much less influenced by their additional contribution). The angle estimated with the Kalman Filter changes less repeatedly and more gradually around the 0 value, representing the vertical axis: the robot can correctly act on the motors and becomes possible to keep it in balance. In conclusion, we set the value of $Q_\theta = 0.001$ and the value of $Q_{\dot{\theta}} = 0.001$.

5.2 The system output

The system output (call it $y(t)$) represents the observation or measurement of the true state $x(t)$, at the instant t. It can be expressed according to the following equation:

[3]For the test, we set the value of $Q_\theta = 0.001$ and the value of $Q_{\dot{\theta}} = 0.001$.

$$y(t) = Hx(t) + v_2(t)$$

where:

- H is a 2x1 static matrix used to map the true state space into the observed space. Since the measurement we are interested in is the angle of inclination computed from the accelerometer measurement, H presents the following structure:

$$H = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

- $v_2(t)$ accounts for the *measurement noise*. It is a random variable with Gaussian distribution of zero mean and covariance matrix R (that we will assume to be constant and independent to the time instant t):

$$v_2(t) = \mathcal{N}(0, R)$$

  Observe that R is associated to a single variable, not a vector. For this reason, the covariance matrix R is in turn reduced to being a single element, coincident with the noise variance. Dwelling on the setting of the variance R, note that if we set it too small the value might be noisy since we trust the accelerometer measurements too much as you can see in Figure 6 where, as usual, the blue line represents the angle proceeds from accelerometer measurement while the red line corresponds to the angle estimated with the Kalman Filter.[4]

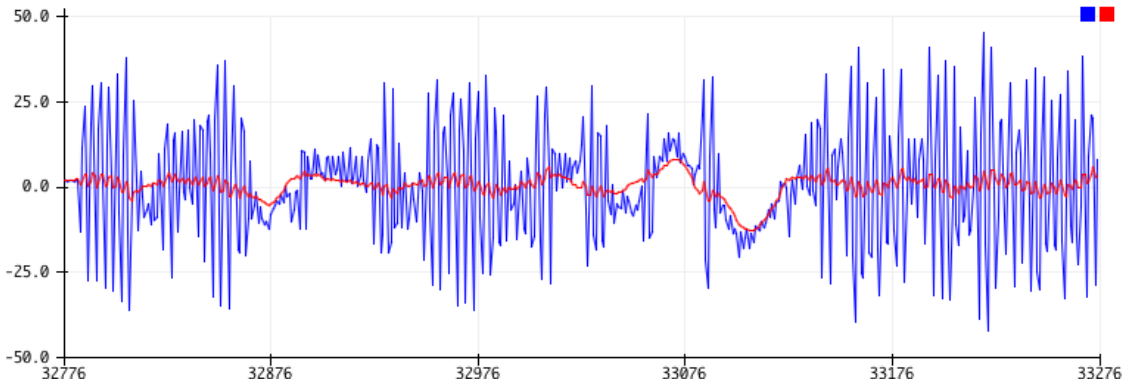


Figure 6: Setting the variance of the *measurement noise* too low the system trust too much the accelerometer measurements. The robot fails to correctly control the motors and soon it falls to the ground.

  The angle estimated using the Kalman filter relies too much on the accelerometer measurement. As a consequence it assumes an oscillatory behavior that the robot fails to manage by operating on the motors and it soon falls on the ground. Conversely, if we set it too high the filter will respond really slowly as it trusts new measurements less, as you can see in Figure 7.[5]

[4]For the test, we set the value of $R = 0.003$.
[5]For the test, we set the value of $R = 3.003$.

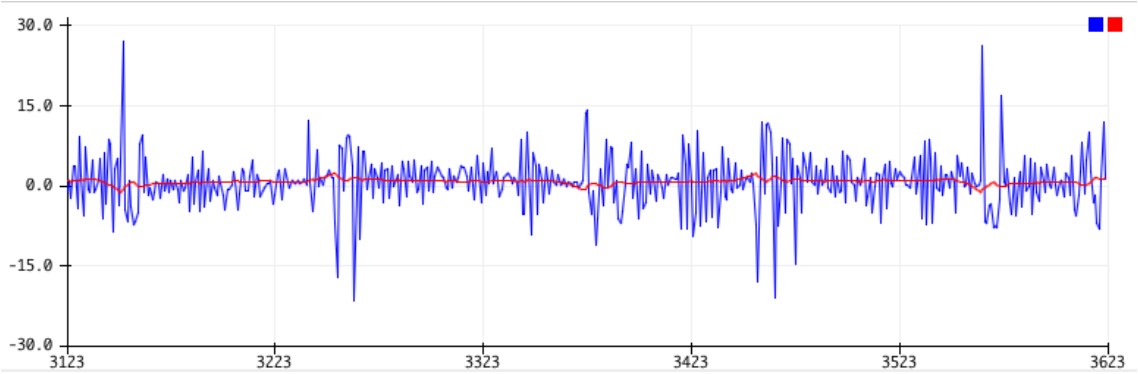


Figure 7: Setting the variance of the *measurement noise* too low, the system does not trust the accelerometer measurements too much. The robot can correctly control the motors and it is possible to keep itself in balance.

However this is the behavior we want to obtain: as previously mentioned, the accelerometer is strongly sensible to the sudden changes of the motors and to the continuous oscillations of the robot that correspond to a great additional noise contribution on the measurement. For this reason, we want the robot does not trust too much the accelerometer measurement to obtain (again) a more fluid and relaxed behavior of the system that has the double benefit of reducing what on the contrary a high value of R creates (i.e. the now known additional noise contributions of the motors), and make the accelerometer measurements more reliable.

5.3 The system model

The final model of our system is therefore represented by the following stochastic system:

$$S : \begin{cases} x(t) = F(t)x(t-1) + G(t)\dot{\theta}(t) + v_1(t) \\ y(t) = Hx(t) + v_2(t) \end{cases}$$

5.4 The prediction phase

In the prediction phase we perform an *a priori estimate* of the system. In other words, the system equations are used to predict the next state and output (call them respectively $\hat{x}$ and $\hat{y}$). Initially, we try to estimate the current state $\hat{x}(t|t-1)$ based on all the previous states $\hat{x}(t-1|t-1)$ and the present gyro measurement $\dot{\theta}(t)$ (i.e. the control input):

$$\hat{x}(t|t-1) = F(t)\hat{x}(t-1|t-1) + G(t)\dot{\theta}(t)$$

Then we try to estimate the a priori error covariance matrix $P(t|t-1)$, based on the previous error covariance matrix $P(t-1|t-1)$:

$$P(t|t-1) = F(t)P(t-1|t-1)F(t)^T + Q(t)$$

The error covariance matrix P is a 2x2 matrix that measures how much we trust the state prediction: it depends on $Q(t)$, so indirectly on the *process noise* $v_1(t)$.
In turn and as previously discussed, Q matrix depends on the interval time Δt between the previous and the current estimation: this makes sense because the more time has elapsed since the last estimate, the more noise is probably introduced (for example, the gyro could have drifted).

5.5 The update phase

After having predicted the state, based on the past values of the system, we execute an *a posteriori estimate* of the system itself. This means we perform the update of the estimate of the state $\hat{x}(t)$ and the covariance error $P(t)$, relying on the current measurement of the output (i.e. the angle registered by the accelerometer).
More in detail, we need to compute at first the prediction of the output $\hat{y}(t|t-1)$, from the previously computed state prediction $\hat{x}(t|t-1)$:

$$\hat{y}(t|t-1) = H(t)\hat{x}(t|t-1)$$

The observation model H is used to map the a priori state into the observed space.
Consequently, we determine the difference between the current value of the angle obtained from the accelerometer, $y(t)$, and its estimation, $\hat{y}(t|t-1)$:

$$e(t) = y(t) - \hat{y}(t|t-1)$$

The resulting term, $e(t)$, is called *innovation* and represents the estimate of the output error. Associated with $e(t)$ we can define an *innovation covariance* matrix $S(t)$[6] that represents how much we should trust the measurement of the accelerometer, based on the a priori state error covariance $P(t|t-1)$ and the measurement variance R:

$$S(t) = H(t)P(t|t-1)H(t)^T + R(t)$$

Again, The observation model H is used to map the a priori error covariance matrix $P(t|t-1)$ into observed space. Note that the higher is the value of the measurement noise variance R, the less we trust the incoming measurement $y(t)$[7].

The next step requires to calculate the Kalman Gain, through the following formula:

$$K(t) = P(t|t-1)H(t)S^{-1}$$

The Kalman Gain, in our case, is a vector composed by two elements and it is used to estimate the contribution of the innovation $e(t)$ in the computation of the elements of the current state vector $\hat{x}(t|t)$: the higher the vector elements of K(t), the more we trust the innovation $e(t)$. We will prove this in a few lines: for the moment, observe that the vector $K(t)$ is inversely proportional to the covariance matrix $S(t)$, therefore it is a logical conclusion to state that when $S(t)$ is great, $K(t)$ is small, and vice versa. Conversely, note that $K(t)$ is directly proportional to $P(t|t-1)$. Finally, take a look at the transpose of the observation model H: it is used to map the state of the error covariance matrix P into observed space.[8]

[6] Observe that, in our case, it coincides with a single scalar value.

[7] This is coherent because S constitutes a matrix of uncertainty in the computation of the innovation $e(t)$, which in turn depends directly on the measurement $y(t)$ (in other words, stating that S measures the uncertainty of $e(t)$ is equivalent to saying that S measures the reliability of $y(t)$).

[8] If the state is not known at startup, it is possible to set the error covariance matrix in the following way: $P(0) = \begin{bmatrix} L & 0 \\ 0 & L \end{bmatrix}$, where L represent a large number. However, in our application we know the starting angle ($\pm 90°$) and we find the bias of the gyro at startup by calibrating, so we assume that the state is known at startup and we initialize the error covariance matrix like a zero matrix: $P(0) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$.

At this point, we can finally refine the *a priori estimate* of the state using the current measurement of the accelerometer $y(t)$ and perform the *a posteriori estimate*, through the following equation:

$$\hat{x}(t|t) = \hat{x}(t|t-1) + K(t)e(t)$$

This formula finally allows us to understand the meaning of the covariance matrices involved in the previous computations. We update the *a priori estimate* by adding an extra term that is directly proportional to the innovation, weighted by the vector $K(t)$. This means we can have two different behaviors, strictly connected to the value assumed by the Kalman gain: $K(t)$ can be a vector with high values or low values in accordance with two different parameters, corresponding to the estimate of the state error covariance matrix $P(t|t-1)$ and the innovation error covariance matrix $S(t)$.
Fixed the latter, if $P(t|t-1)$ is a matrix with small values, it is reasonable to believe we are trusting a lot the *a priori estimate* $\hat{x}(t|t-1)$. As a consequence, $K(t)$ results in a small vector (remember that $K(t)$ is directly proportional to $P(t|t-1)$) and the additional term $K(t)e(t)$ becomes negligible so that the *a posteriori estimate* $\hat{x}(t|t)$ is approximately equals to the *a priori estimate* $\hat{x}(t|t-1)$. In other words, we trust the *a priori estimate* so much that it is not necessary to update the prediction with the contribution offered by the current measurement. Instead, if P(t|t-1) is a matrix with sufficiently big values, the contribution of $K(t)$ becomes relevant and the term K(t)e(t) is no more negligible so that the *a posteriori estimate* $\hat{x}(t|t)$ is equals to its formal equation $\hat{x}(t|t-1) + K(t)e(t)$. This means we do not trust the *a priori estimate* so much that it is necessary to consider the contribution provided by the current measurement, in terms of innovation, in order to update the prediction.
In the same way, fixed the state error covariance matrix $P(t|t-1)$, if $S(t)$ is a small scalar, then we have reason to believe that the computed innovation is reliable (has a low margin of variability). In this case, $K(t)$ becomes significant (remember that $K(t)$ is inversely proportional to $S(t)$) and this applies accordingly also for the term $K(t)e(t)$, in the *a posteriori estimate* formula. On the other hand, if $S(t)$ has a big value, we do not trust the current measurement and the corresponding innovation: $K(t)$ is a small vector and the term $K(t)e(t)$ becomes negligible at the point that the *a posteriori estimate* tends approximately to the *a priori estimate*.

To recap:

- fixed the matrix $S(t)$, if:

  1. $P(t|t-1) \to \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \Rightarrow K(t) \to \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and $\hat{x}(t|t) \simeq \hat{x}(t|t-1)$
  2. $P(t|t-1) = \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}$ (with $P_{00}, P_{01}, P_{10}, P_{11}$ suff. big) $\Rightarrow$
     $\Rightarrow K(t) \begin{bmatrix} K_0 \\ K_1 \end{bmatrix}$ (with L_0, L_1 suff. big) and $\hat{x}(t|t) = \hat{x}(t|t-1) + K(t)e(t)$

- fixed the matrix $P(t|t-1)$, if:

  3. $S(t) \to 0 \Rightarrow$
     $\Rightarrow K(t) = \begin{bmatrix} K_0 \\ K_1 \end{bmatrix}$ (with L_0, L_1 suff. big) and $\hat{x}(t|t) = \hat{x}(t|t-1) + K(t)e(t)$

4. $S(t) = L$ (with L suff. big) $\Rightarrow K(t) \rightarrow \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and $\hat{x}(t|t) \simeq \hat{x}(t|t-1)$

In the end, it is necessary update the state error covariance matrix to the current measurement. Formally speaking, we must calculate $P(t|t)$, that is obtained through the following equation:

$$P(t|t) = (I - K(t)H)P(t|t-1)$$

where I represents a 2x2 *identity matrix*.
More in detail, developing the equation,

$$P(t|t) = (\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} K_0 \\ K_1 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix}) \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}$$

Focusing the attention on the first term $(I - K(t)H)$, which represents the updating factor of the *a priori estimate* of $P(t|t-1)$, we can notice that if the gain vector $K(t)$, provided by the new measurement $y(t)$, is a vector with small modulus, consequently $\mid I - K(t)H) \mid \rightarrow 1$ and the *a posteriori estimate* is approximately equal to the *a priori estimate* ($P(t|t) \simeq P(t|t-1)$). Conversely, if the gain vector $K(t)$ is a vector with high modulus, $\mid I - K(t)H) \mid \rightarrow L$ (with L sufficiently big) so that the correction on the *a priori estimate* is significant.

6 Motor Control

Once the filtered value for the angle is obtained from Kalman we need to perform the appropriate correction in order to maintain the stability of the robot. We have several problems concerning the control of the two mounted brushed motors:

- Offset in the activation voltage needed to actually make the rotor spin; it is different between the two motors and can be greatly reduced by a one-time calibration procedure.
- Significant backlash, due to the toy-grade transmission used. It is an error that occurs in gear systems that change direction of movement and it's due to excessive clearance in joints. This is the most impacting problem we have to face, since the robot needs to continuously invert wheels rotation in order to stay in equilibrium.
- Non linearity in the speed settings, caused not only by the motors themselves but possibly also by the driver board.

A PID Controller is responsible for controlling the motors. We have found the optimal settings in an high integral response, a quite high proportional contribution and a null derivative component, thus configuring the controller in its PI version (see Figure 8)[9].

The *derivative* component introduces too much nervousness in the response because the sudden actuation of the motors misleads the accelerometer with an unwanted contribution. For this reason it's not accounted for the control of the system.

[9]We have found the best settings in $k_p = 150$ and $k_i = 300$, where k_p is the proportional coefficient and k_i the integral one. These values have been empirically derived; we could not apply the well proven Ziegler–Nichols method because it was impossible to get the stable and consistent oscillations required.

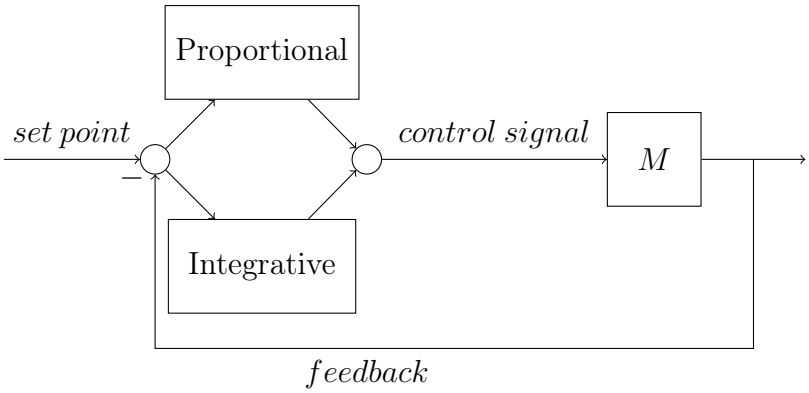


Figure 8: PI Controller Block Diagram

The *integral* part helps a lot in the stabilization of the robot because its effect grows up over time without sudden changes; moreover it avoids the robot to make a continuous movement in a single direction, at least at some extents, filling the lack of motor encoders. Since the integral term has memory of the past behavior of the system, it is necessary to limit it in order to prevent the problem called windup. This problem arises when the error stays fixed in one direction for a long period of time, thus allowing the integrator to accumulate an increasingly high correction.

The *proportional* part, in the end, gives the major contribution to the standing ability of the robot and it is considerably big because, due to the reduced power and torque of the motors, the robot cannot stand too high deviations from the vertical equilibrium.

7 Conclusion

In conclusion, we report a comparison performed by first starting the robot without the Kalman filter, then introducing the implemented filter. The comparison aims to reinforce and highlight the importance of the work done.

As it is possible to see in Figure 9, the behavior of the system, based on the measurement of the angle provided by the accelerometer (blue line) and manipulated by the microcontroller without any filtering, is absolutely instable.

The action of the PI controller (red line) on the motors is very oscillating and contributes to make the measurement of the current incline even more uncertain, rendering the robot nervous and uncontrollable. The robot remains standing only because we try to balance it, acting with our hands and bringing it back for an instant into the equilibrium condition: from the moment we stop supporting the robot with our contribution, the robot quickly loses control and falls to the ground (as it is possible to see on the extreme right of the figure).

Conversely, the behavior of the system when the Kalman filter is applied to the measurement of the accelerometer is positively stable and controllable, as you can see in Figure 10 where the blue line represents the filtered angle of the robot and the red line the action of the PI controller on the motors. Unlike the previous figure, the measured angle oscillates on the neighbourhood of the zero (the vertical axis): the control on the

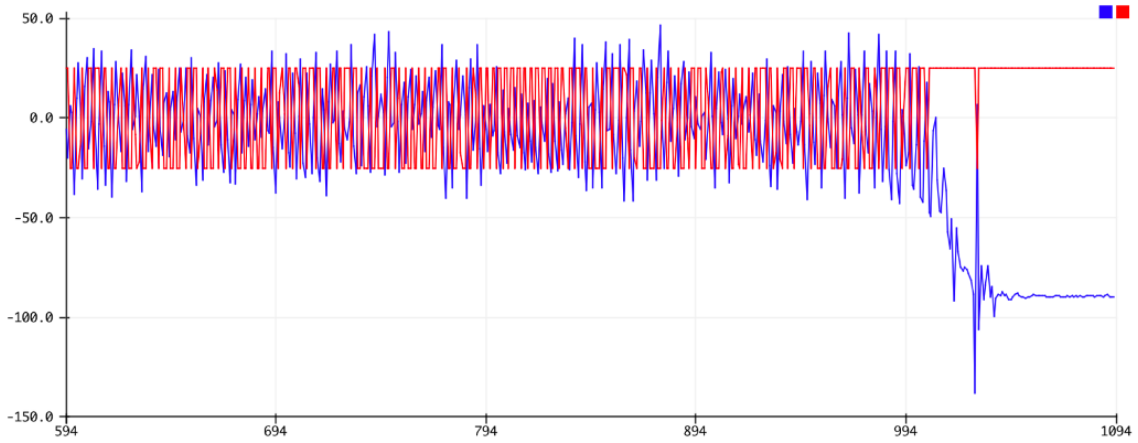


Figure 9: The robot acts on the motors according to the angle registered by the accelerometer, without performing any filtering on the measurement. The robot is unstable and immediately falls down to the ground.

engines is not reactive, but progressive, and this helps to make the accelerometer less subject to the additional noise (due to the action of the motors) and more reliable in the next measurement. The robot finally manages to keep itself in balance, in total autonomy.

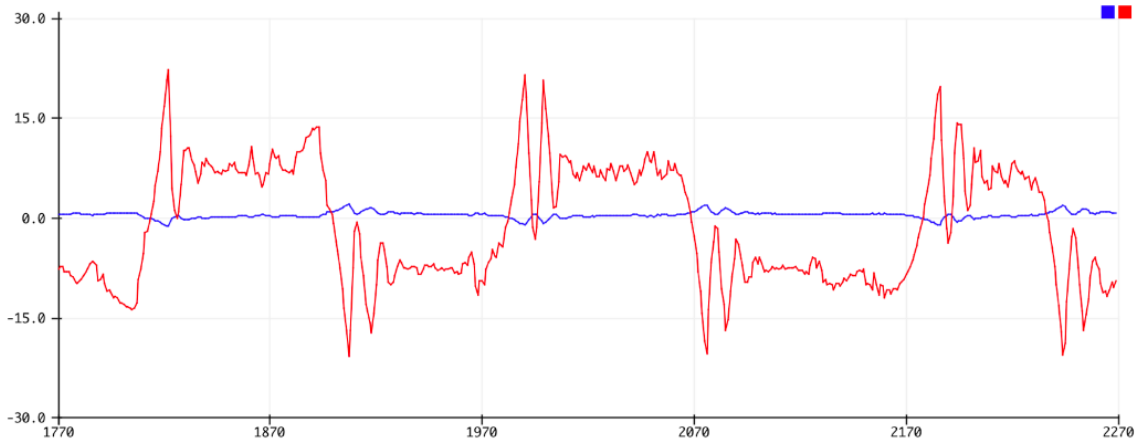


Figure 10: The robot acts on the motors according to the angle registered by the accelerometer and filtered by the Kalman filter. The system is stable and manages to keep the robot in balance.

References

[1] Sergio Bittanti. *Teoria della predizione e del filtraggio*. Pitagora Editrice, Bologna, 2002.

[2] Piroddi Luigi. *Kalman Filtering*. Proprietary notes, Politecnico di Milano, 2017.

[3] Starlino Electronics. *A Guide To using IMU (Accelerometer and Gyroscope Devices) in Embedded Applications*, 2010. [Online]. Available: `http://www.starlino.com/imu_guide.html`. [Accessed: 22-03-2017].

[4] I2Cdevlib. *MPU6050 Class Reference*, 2011. [Online]. Available: `https://www.i2cdevlib.com/docs/html/class_m_p_u6050.html`. [Accessed: 22-03-2017].

[5] Kristian Sloth Lauszus, TKJ Electronics. *A practical approach to Kalman filter and how to implement it*, 2012. [Online]. Available: `http://blog.tkjelectronics.dk/2012/09/a-practical-approach-to-kalman-filter-and-how-to-implement-it/`. [Accessed: 22-03-2017].

[6] Samuel Walsh. *Processing data from MPU-6050*, 2014. [Online]. Available: `http://samselectronicsprojects.blogspot.it/2014/07/processing-data-from-mpu-6050.html`. [Accessed: 22-03-2017].

[7] Giuseppe Caccavale. *MPU-6050 (GY-521) Arduino Tutorial*, 2015. [Online]. Available: `http://www.giuseppecaccavale.it/arduino/mpu-6050-gy-521-arduino-tutorial/`. [Accessed: 22-03-2017].