FRTN01 Project: Stabilization of MinSeg

Description of the MinSeg

The MinSeg (`https://minseg.com`) is a miniature balancing robot, specially developed for teaching in control courses. By using readings from the on-board inertial measurement unit (IMU), it is possible to estimate the angle and angular rate of the robot, and use feedback control to steer the wheel motor and balance the robot upright. Since the MinSeg only has one motor, it is limited to driving along a line.

You will use the kit known as the *MinSegShield M1V4-kit*, which consists of:

- An Arduino-compatible microcontroller (SainSmart Mega 2560).
- A MinSegShield M1V4, mounted on the microcontroller.
- A Lego NXT DC motor, with an encoder.
- An IMU with 3-axis accelerometer and 3-axis gyroscope.
- 9V AA battery box (uses 6 batteries).
- A USB adapter for communication between the MinSeg and computer.

A full list of components and pin-out diagram for the MinSegShield can be found on the MinSeg website.

The shield also have a bluetooth header, which makes it possible to plug in a bluetooth module and implement wireless communication between the MinSeg and the computer.

To reduce the overhead of designing and implementing a controller for the MinSeg, the developers have created a software library called *RASPLib* which makes it possible to use Matlab/Simulink for implementation. However, in this project your final implementation should be written from scratch in Arduino-code (although you are encouraged to use RASPLib for trying out the control algorithm beforehand). Arduino-code is based on C, and is easy to develop using Arduino's own IDE, which makes it simple to upload code to the micro-controller and print data. If you are new to Arduino, then you are recommended to look throuhg the guide here: `https://www.arduino.cc/en/Guide/HomePage`.

Project Goals

The goals of this project are:

1. To design and implement an estimator/sensor fusion and a controller that is able to balance the MinSeg in an upright position.
2. Being able to receive and plot data in real-time from the MinSeg on a computer.
3. Being able to send simple commands and/or change of controller parameters to the MinSeg from the computer.

Ideally, your final implementation should use wireless communication via bluetooth.

Hints for Getting Started

Here are some hints for getting started.

RASPLib

There is a link for download of RASPLib at the MinSeg website. The .zip-file contains instructions on how to install RASPLib and install drivers on the MinSeg.

Even if you should not use RASPLib for your final implementation, it can provide a good starting point for your control design. It should make it easier to get started, and if the control is not satisfactory you can be quite certain that it is due to the design itself, rather than some bug in your code.

Also, you might find it useful to "peek under the hood" of how the Simulink blocks in RASPLib are implemented, e.g. how the interfaces to IMU and motor are implemented. By clicking on a block and choosing "see source code", you should be able to see the underlying C-code implementation. All the C-code is also located under `RASPLib/src` and `RASPLib/include`.

The IMU

An IMU is a sensor unit which contains both a gyroscope and an accelerometer. The gyroscope measures angluar rate in 3 dimensions (roll, pitch, yaw), and in principle one could simply integrate the measurement directly to obtain the angle. However, there is often a small bias in the gyro measurements, which will lead to a drift after integrating the signal. This can be interpreted as the gyroscope having unreliable measurements for low frequencies, and we should therefore only use the high-frequency part of the measurements.

The accelerometer measures the acceleration in 3 dimensions *relative to free fall.* This means that when the accelerometer is not moving, it will measure an acceleration of 9.82 m/s^2 straight up from the ground (it would have been 0 m/s^2 in free fall). Unlike the gyroscope, the accelerometer does not have the same problem with drift, but instead have quite noisy measurements. To get reliable measurements, we need to average them over a long time-span. This can be interpreted as the accelerometer having unreliable measurements for high frequencies, and we should therefore only use the low-frequency part of the measurements.

An estimator/sensor fusion algorithm which estimates the angle of the MinSeg should thus use the properties of these two sensors to its advantage. For your implementation, I would recommend either using a complementary filter (simple, no model required, possibly a bit crude) or a Kalman filter with a simple noise model (requires a model, but better performance).

The IMU on the MinSeg is an MPU6050 from Invensense, which uses I2C to communicate with the Arduino board. There is a library by Jeff Rowberg available for (among others) Arduino (`https://github.com/jrowberg/i2cdevlib`) that takes care of the I2C communication for you, and should make it simpler to use the IMU. The library specific for MPU6050 is found here:

`https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050`

You also need the general I2C library (the MPU6050 code uses it) found here:

`https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/I2Cdev`

Once you have added the two libraries above to your Arduino IDE, you can check the examples for the MPU6050 library. The two interesting examples are `MPU6050_raw` and `MPU6050_DMP6`. The first example shows how to run a loop on the Arduino, where you poll the IMU for raw measurements at each iteration. The second example uses the more sophisticated *digital motion processor* (DMP) on the IMU, which can do some pre-processing of the data (e.g convert the data into human-readable Euler angles) before putting it in a FIFO buffer. Each time the IMU puts a new value in the buffer, it signals an interrupt to the Arduino, which then can fetch the new data. However, this mode of operation requires that you connect the port labeled `INT` on the IMU to a digital pin with interrupt on the Arduino board. One walkthrough (there are several) for getting the libraries above and running the `MPU6050_DMP6` example can be found here:

https://www.instructables.com/id/MPU6050-Arduino-6-Axis-Accelerometer-Gyro-GY-521-B/

Also, here is an example for calibrating the IMU measurements:

http://forum.arduino.cc/index.php?action=dlattach;topic=397918.0;attach=206004

The Motor

The motor is a Lego NXT DC motor, which generates a torque by inputting a PWM signal from the Arduino via the motor header on the shield. It features an internal encoder which measures the rotation of the motor shaft. The encoder uses 12 holes with a so called quadrature design, meaning it has a resolution of 48 pulse changes per revolution. The motor shaft is connected via a gear with transmission ratio 15, meaning the encoder will deliver $48 \cdot 15 = 720$ pulse changes per motor shaft revolution, leading to a maximal theoretical accuracy of $0.5°$. By counting the number of pulses each period of your control algorithm, you can thus compute the angular rate of the wheels, from which you can compute the horizontal velocity of the MinSeg. By simply keeping a running sum of pulses (with sign depending on motor direction) over all sampling period, you can compute an estimate of the MinSeg's position. Note that this position estimate will likely start to drift after a while, but for basic implementations that is ok.

To run the motors using Arduino, you can take inspiration from the underlying implementations of the motor blocks in RASPLib. Also, the following library could perhaps be of use:

https://github.com/sebgiles/NXTServo

Using Bluetooth

On the shield of the MinSeg there is a header ready for plugging in a bluetooth module. The module that you will use is a HC-06 or HC-05. The difference between these two modules is that HC-06 only works as a slave, i.e. it cannot connect to other devices on its own (e.g it cannot establish connection with another MinSeg), whereas the HC-05 can. Depending on how you wish to implement your project, the choice of HC-05 or HC-06 could therefore matter.

A tutorial for using the HC-06 with Arduino can be found here:

https://www.aranacorp.com/en/arduino-and-bluetooth-module-hc-06/

Use a Model When Designing Your Controller

While you are exploring the hardware and code, make sure that you in parallel start developing a mathematical model describing the dynamics of the MinSeg. This enables you to use common design methods such as pole-placement, LQG et.c, and it helps to bring insight to what is required by the controller. There are many models for segway robots/inverted pendulums online, if you need inspiration.

USB- vs. Battery-Powered

The MinSeg can be powered both by batteries and through a USB connection. While the USB connection is convenient, the motors will only be supplied with 3.25V. With batteries however, the voltage is 9V. This means that the motors will be more powerful when using batteries than when using USB, and it may thus be easier to control the MinSeg. The control also becomes a bit easier with the batteries, since they raise the center of mass of the MinSeg. However, it is of course still possible to control the MinSeg using USB! It is just something that you need to take into account when you design your controller.