Custom Two-wheeled Inverted-pendulum Hardware Platform

Motivation

In my master thesis I developed a networked control system for a Two-wheeled Inverted-pendulum robot (TWIPR) based on the popular Lego Mindstorms EV3 platform. The Lego Mindstorms platform suffers from a variety of disadvantages. E.g. the position sensor, which is crucial for a balancing robot, is of bad quality, since it only contains a 1-d gyroscope. Furthermore, the ARM-based processor of the EV3 brick is quite outdated and requires an external WiFi dongle for wireless communication. And to put the cherry on the cake: the Lego set required to build a TWIPR costs around 400€.

Since TWIPR are a popular topic in control theory teaching and research, I have decided to build a custom TWIPR hardware, which does not suffer from the above-mentioned drawbacks and costs only a fraction of the Lego set. All the hard- and software is published open source so that hopefully others can use and extend this work.

Hardware

The core component of the hardware platform is the Raspberry Pi 3B+, which is the most recent version of this popular compact computer. It is equipped with a modern ARM-based processor and comes with all the peripherals needed for this project, like an on-board WiFi chipset.

As said before, one of the major drawbacks of the Lego platform is its bad gyroscopic sensor. Therefore, I am using a fully featured MPU6050 based inertial measurement unit (IMU) for my custom setup. These chips come with a full 3-axis accelerometer and a 3-axis gyroscopic sensor, which allows to employ sensor fusion for the calculation of the robot pitch angle, delivering much more accurate results than the Lego gyroscope.

A TWIPR is obviously supposed to move around and therefore needs motors. I have chosen DC-motors with a metal gear and an integrated quadrature encoder, which enables the measurement of the motor shaft angle with a precision of half a degree. The motors cannot be driven by the Pi directly, so it is necessary to use motor driver chips. In the case at hand TI's DRV8870 chips were chosen, since their specs match the ones of the employed motors.

Since the frequency of the encoder signals can get quite high with high motors speeds, I have decided not to read the encoder signals directly with the Pi, but to use ATTINY85 microcontrollers for that. For the prototype Digispark ATTINY85 USB development boards were used.

For the power supply eight standard 1.2V NiMH rechargeable batteries are used. They deliver a voltage between 9V and 11V depending on the state of charge. Since the Pi needs a supply voltage of 5V and it is not desirable to supply the motors with a variable voltage, DC-DC step-down converters are needed. In the case at hand LM2596 based DC-DC converter boards are used, which can supply a current up to 3A. Three converters are used in total, one supplying 5V for the Pi and one that supplies 8V for each motor. The IMU, the ATTINY85 boards and the motor encoders are supplied by the Raspberry Pi.

The following list summarizes all the required hardware components:

- Raspberry Pi 3B+
- MPU6050 based IMU board
- 2x DFRobot Fit0521 geared DC motors
- 2x Texas Instruments DRV8870 motor drivers
- 3x LM2596 based step-down DC-DC converter boards
- 2x Digispark ATTINY85 USB Development boards (or a compatible version)
- 2x 47uF electrolyte capacitors
- 2x 0.1uF ceramic capacitors
- 4x 4.6k pull-up resistors
- AA Battery older for 8 batteries
- Rocker switch, connectors and cables

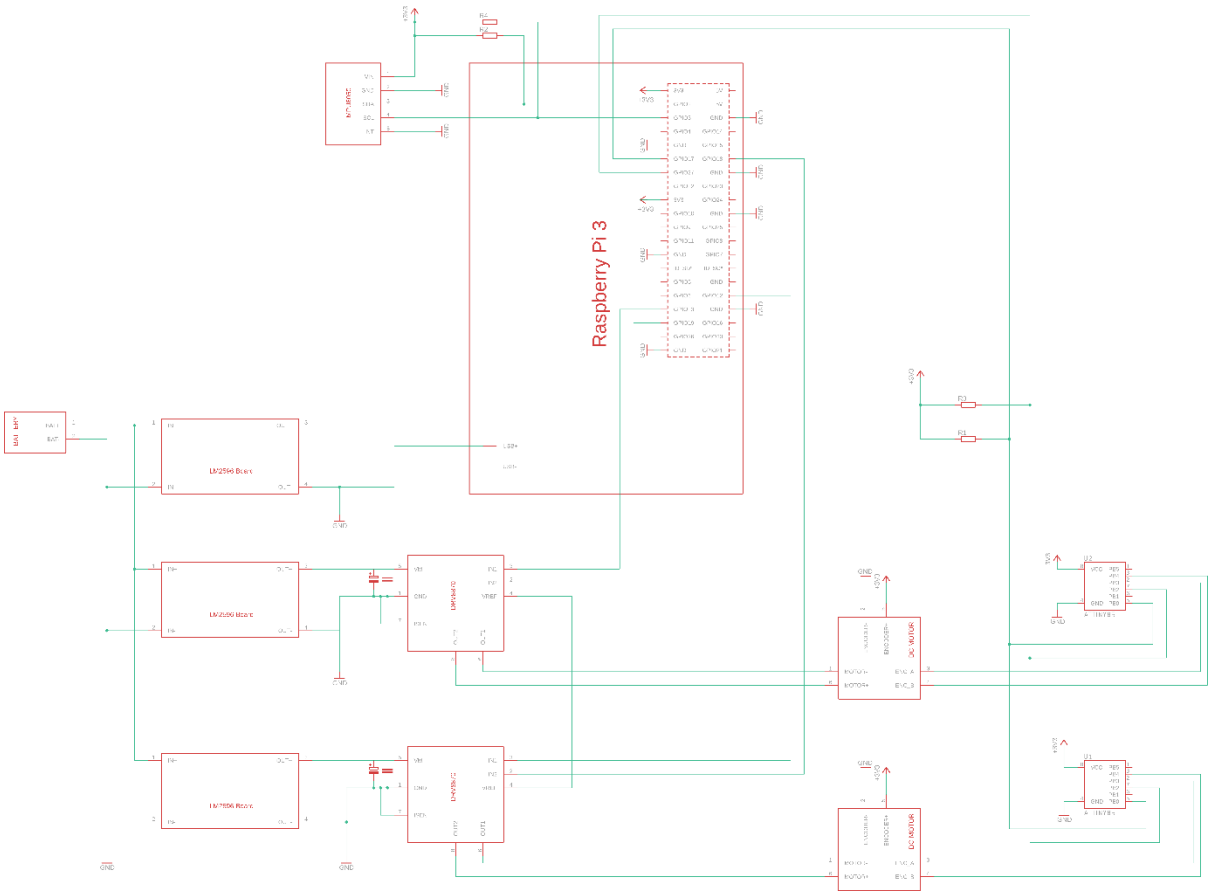
Schematics

The interconnection of the components is quite straightforward. But there are two things that need to be considered. The motor speed is regulated using a PWM. Therefore the inputs of the motor drivers have to be connected with appropriate PWM pins of the Pi. The seconds issue is related to the I2C communication between the Pi, and the MPU6050 and ATTINY85 boards.

ATTENTION: Due to a hardware bug the Pi's hardware I2C does not allow for "clock stretching". The MPU6050 is fast enough, so it does not need that feature, but the ATTINY85s are not. Therefore, software I2C is used for the communication with the ATTINY85 boards.

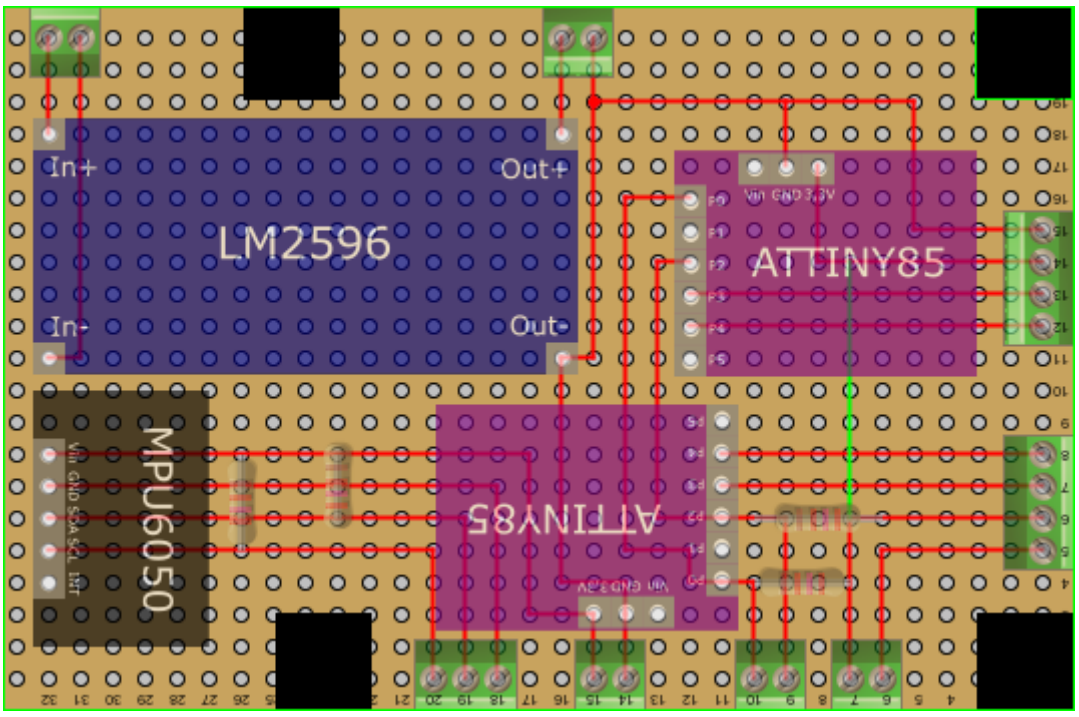
So the SDA/SCL pins of the MPU6050 are connected with the hardware SDA/SCL pins of the Pi (GPIO2/GPIO3). The SDA/SCL pins of the ATTINY85 boards are connected with arbitrary GPIO pins of the Pi, which are configured as the SDA/SCL pin of the software I2C.

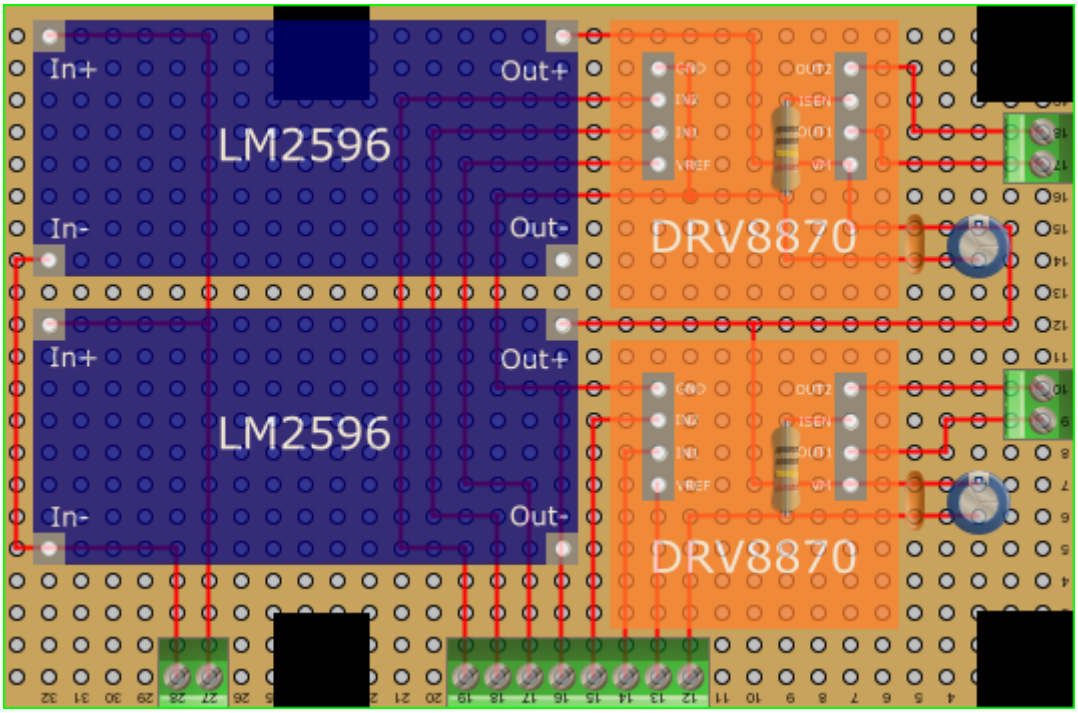
A full EAGLE schematic of the project is provided and should answer all open questions.



Prototype

While a proper layouted PCB is desirable, the prototype was built using perfboards.

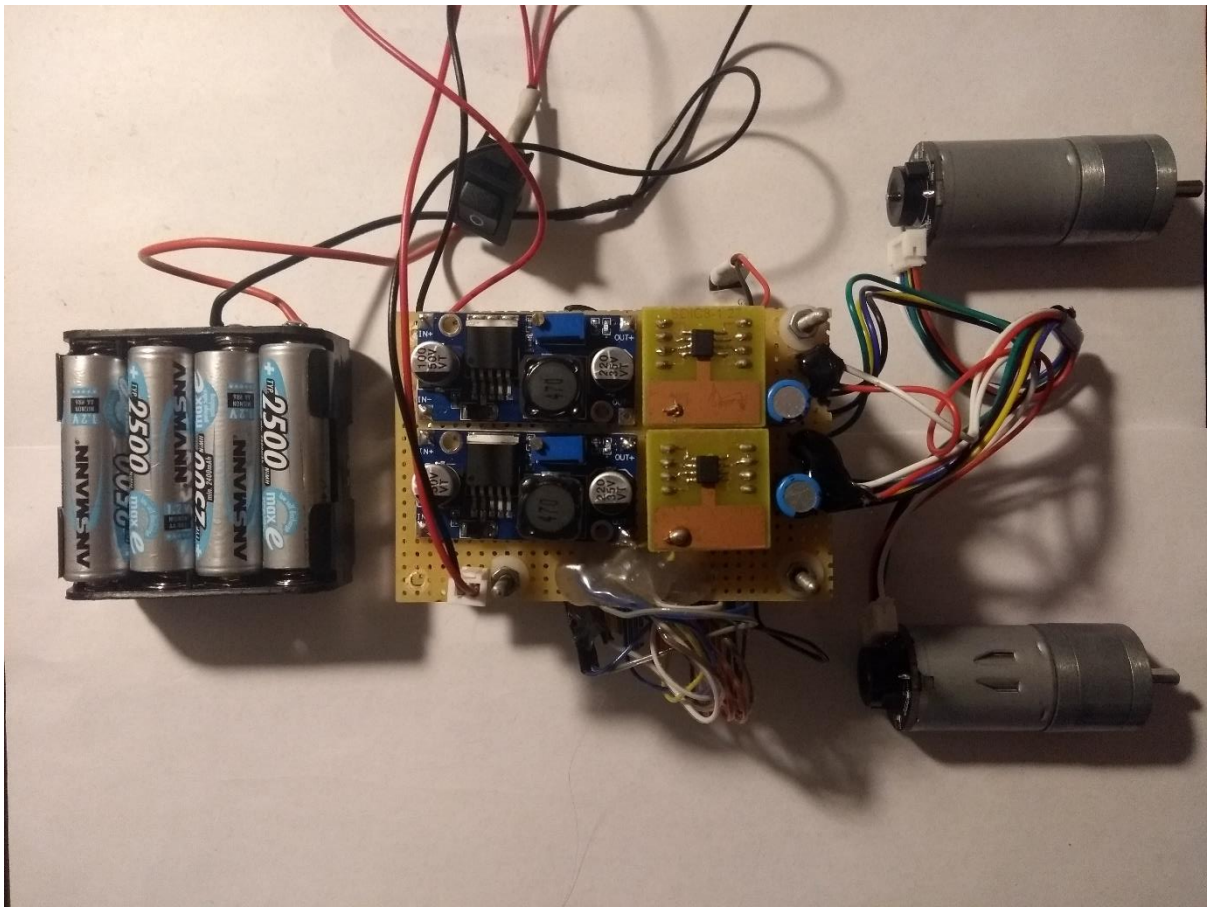


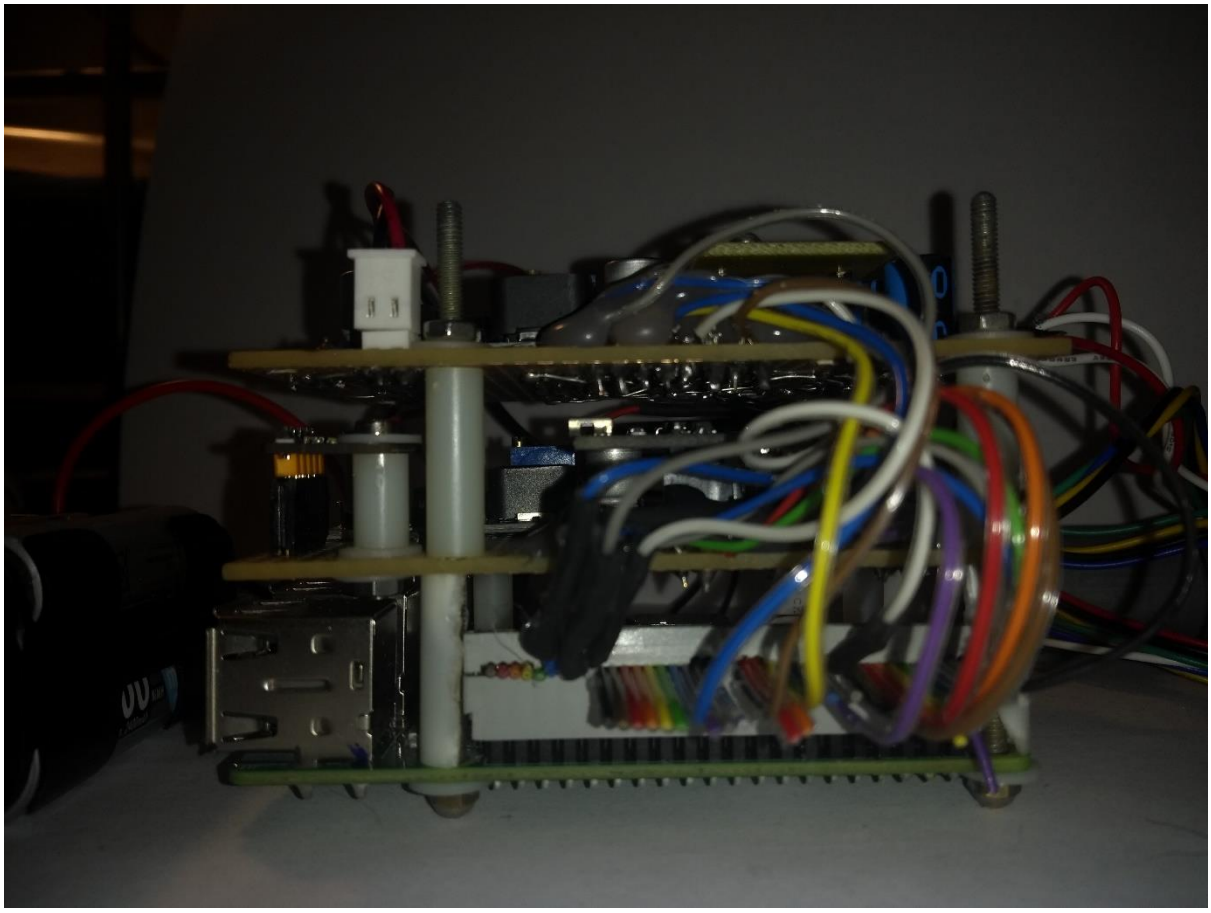


The perfboards have the same size as the Raspberry Pi. Therefore, they can be stacked to form one compact hardware compenent.

The above shown images were created using the BlackBoard Circuit Designer. These files are provided as well.

The complete setup:





Software

Different software modules for the use of the motors, encoders and IMU were implemented in Python.

Please note: The Python modules require Pigpiod. More information: http://abyz.me.uk/rpi/pigpio/pigpiod.html

motor_drive.py: Module in which the functions to regulate the motor speed are implemented.

Functions: **self.stop_motors( )**

The PWM duty cycles of both motors are set to 0.

self.brake_motors( )

Active breaking.

self.set_duty(left_duty, right_duty)

Sets the motor voltages to the provided values.

left_duty: Float, left motor duty cycle. Min: -100 (full speed, counterclockwise) Max: 100(full speed, clockwise)

right_duty: Float, right motor duty cycle. Min: -100 (full speed, counterclockwise) Max: 100(full speed, clockwise)

left_duty, right_duty = self.get_duty( )

Returns the currently applied duty cycles.

motor_encoder.py: Module in which the functions to read the motor angles from the ATTINY85 boards are implemented.

Functions: **angle_l, angle_r= self.get_motor_angles( )**

Returns the current motor angles.

IMU_class.py: Module in which the functions to read pitch angle and angular velocity from the IMU are implemented.

Functions: **pitch_velocity, pitch_angle= self.get_IMU_values(ts)**

Returns the current pitch angular velocity and pitch angle. Parameter: Sample time.

ATTINY85_encoder.ino: Arduino code for the ATTINY85 board. Please note: One board should be programmed with SLAVE_ADDR = 0x40 and the other one with SLAVE_ADDR = 0x48.