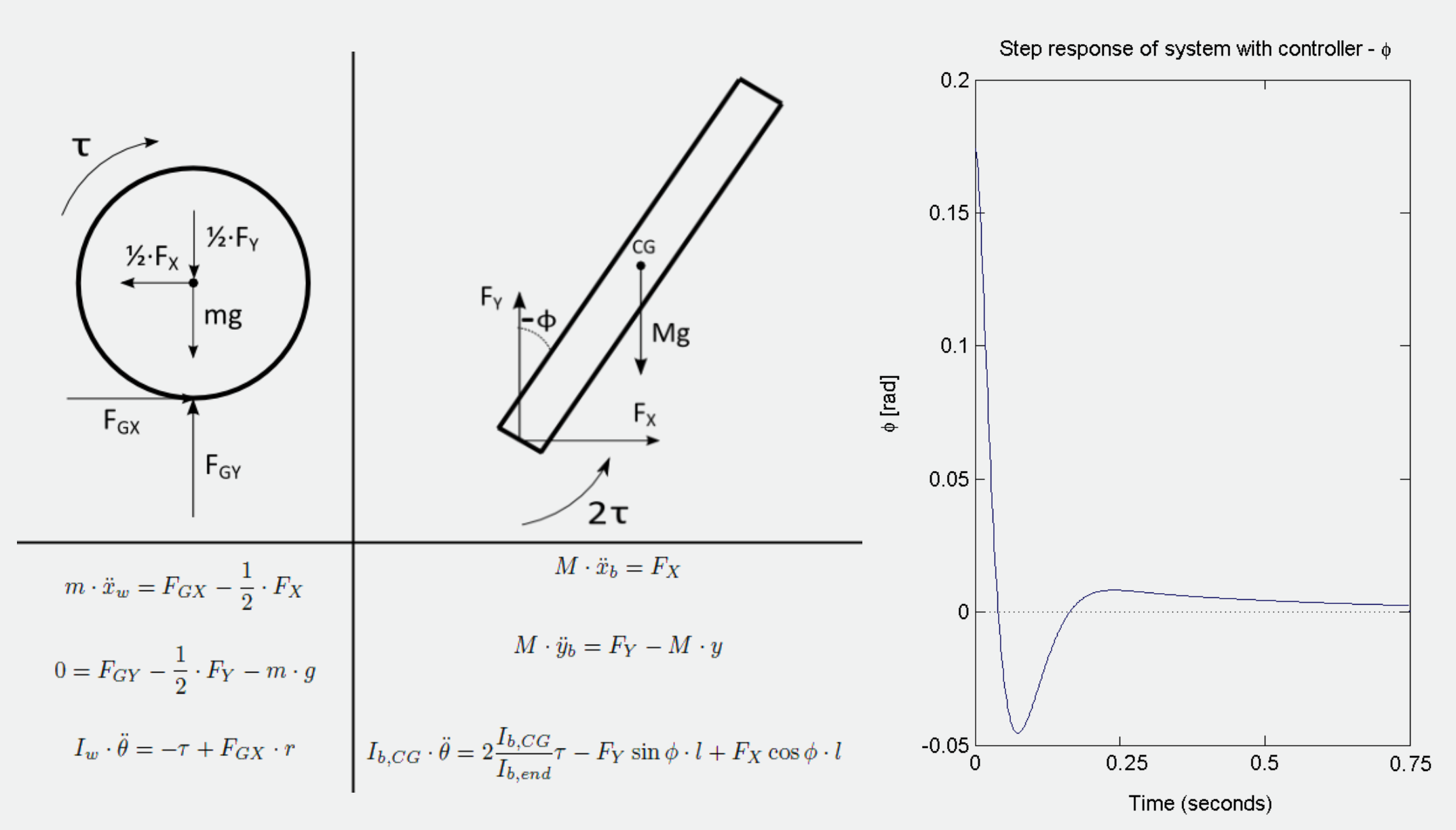
Two Wheeled Balancing Robot

Mads Bornebusch (s123627)   Kristian Lauszus (s123808)

Abstract The goal of the project is to build a two wheeled self-balancing robot capable of carrying a person. The dynamics of the robot is found and the system is simulated and stabilized with a controller in Matlab. The robot is programmed in C/C++ on an 8-bit ATmega328P AVR microcontroller. A manually tuned PID-controller is used to keep it balanced. The robot is able to communicate with an Android device over a two way Bluetooth connection.



$$m \cdot \ddot{x}_w = F_{GX} - \frac{1}{2} \cdot F_X$$

$$0 = F_{GY} - \frac{1}{2} \cdot F_Y - m \cdot g$$

$$I_w \cdot \ddot{\theta} = -\tau + F_{GX} \cdot r$$

$$M \cdot \ddot{x}_b = F_X$$

$$M \cdot \ddot{y}_b = F_Y - M \cdot y$$

$$I_{b,CG} \cdot \ddot{\theta} = 2\frac{I_{b,CG}}{I_{b,end}}\tau - F_Y \sin\phi \cdot l + F_X \cos\phi \cdot l$$

Fig. 1 – Model and simulations Using the free body diagrams to the left, the equations for the robot is set up (left bottom). These lead to a transfer function from τ to ϕ for the robot and a controller can be designed in Matlab. The controller is designed by choosing ω_c which is the cross frequency of the Bode plot of the system. The step response with the controller inserted is shown to the right. It can be seen that it is indeed possible to stabilise the system.

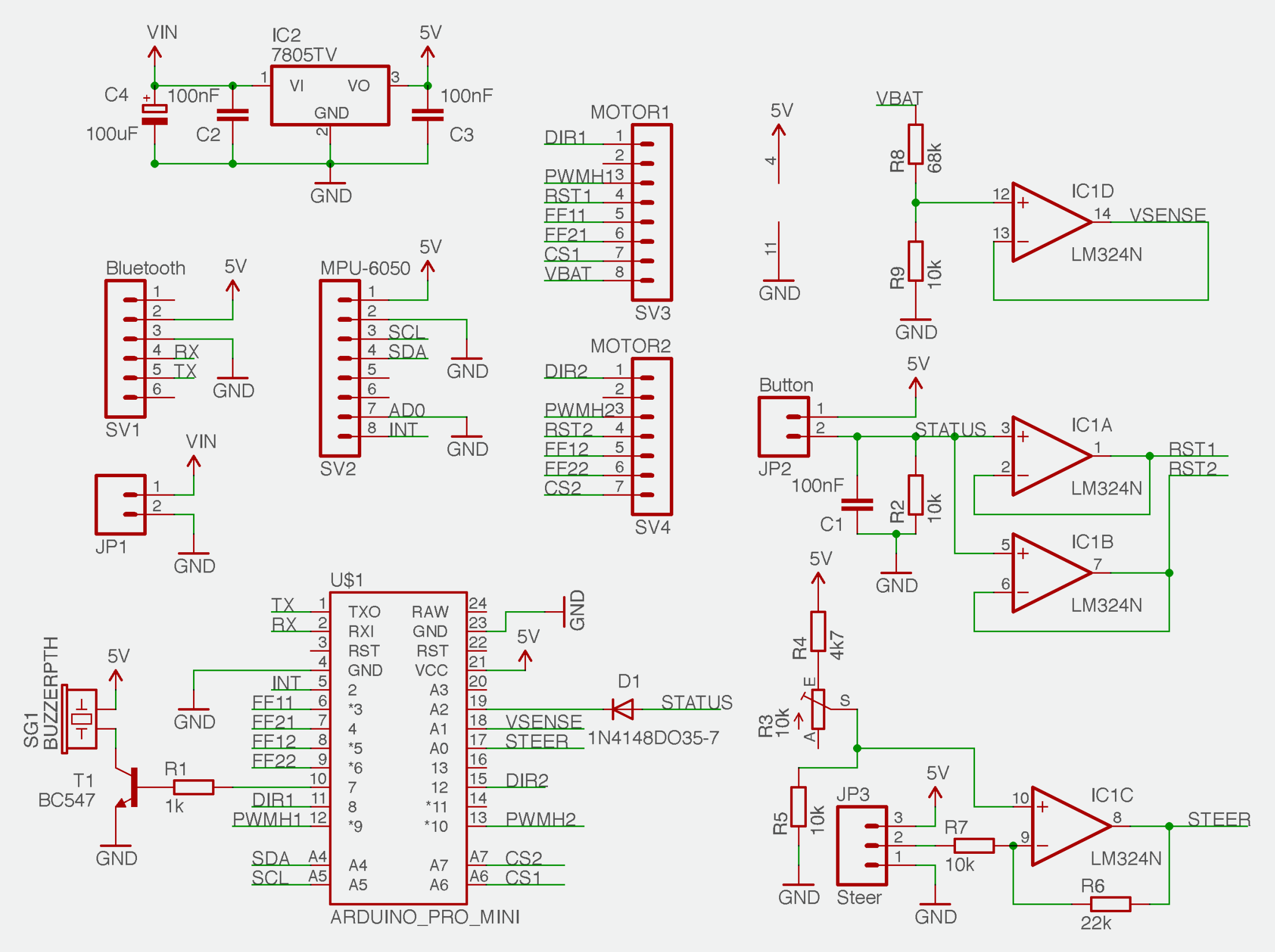


Fig. 2 – The Hardware The circuit diagram in the figure shows the circuit on the main board of the robot. This contains the microcontroller controlling the robot and is connected to the motor drivers which controls the motors.

Important components on the main board:

- Arduino Pro mini
  The microcontroller on this board is an 8-bit ATmega328P AVR microcontroller running at 16MHz.
- MPU-6050 Inertial measurement unit (IMU)
  The microcontroller is reading from the IMU at 500Hz. The range for the accelerometer is set to $\pm 2g$ and the range for the gyroscope is set to $\pm 250°/s$.
- Bluetooth SPP module
  This sends serial data from the microcontroller to the Android application via SPP.
- LM324 Operational Amplifier
  The Op-Amp is used as a buffer and an amplifier for the output of the potentiometer used for steering.
- 7805 Linear 5V regulator
  This regulator is used to supply power to the components of the main board. It is supplied by the LM2596 (see below).

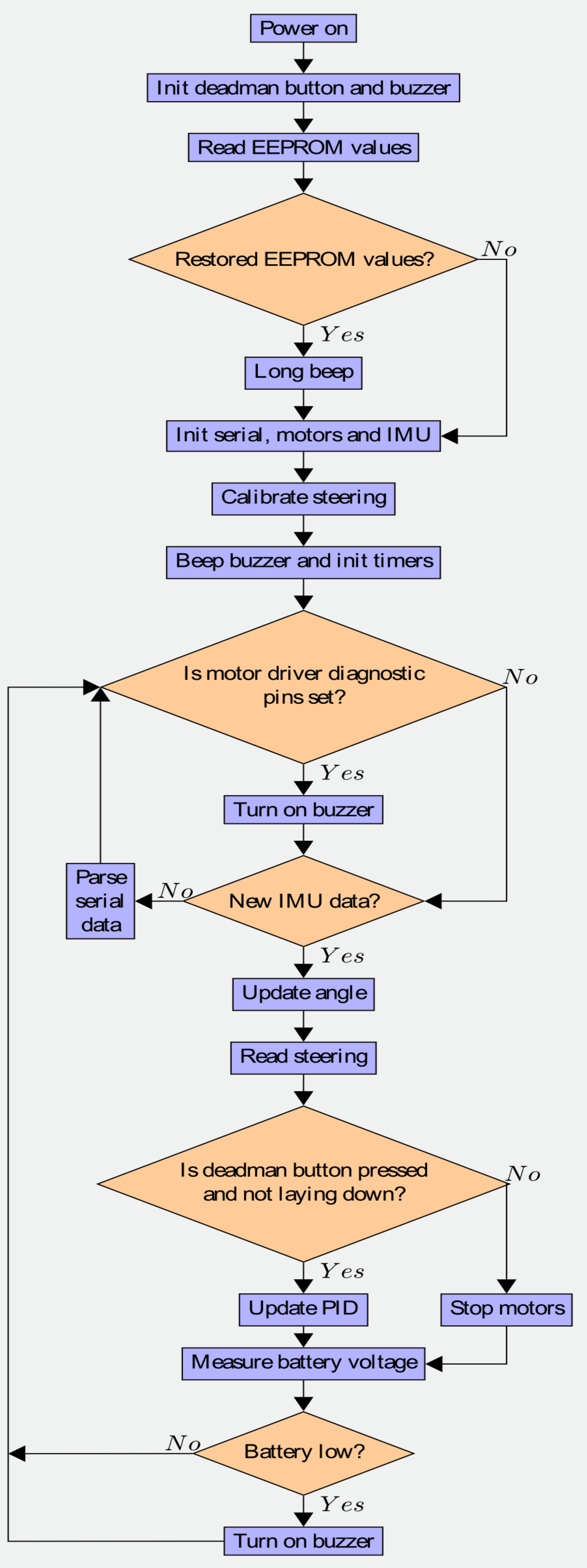
Other important components (not on main board):

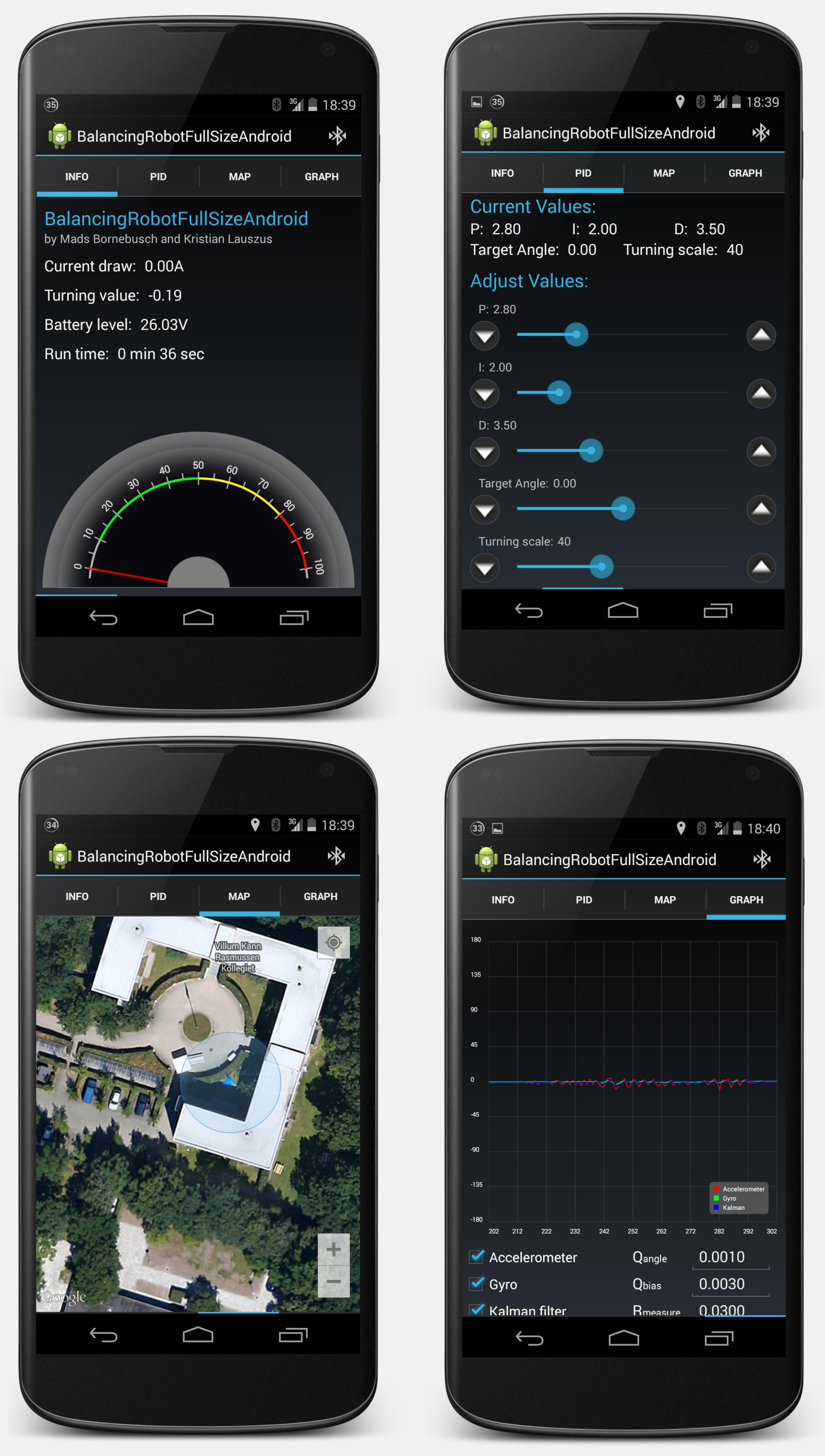
- LM2596 Adjustable Switching Regulator
  This regulator is used to step down the battery voltage to 8V to supply the 7805. This will minimize the power loss.
- 10 kΩ Potentiometer
  The potentiometer measures the angle of the steering rod.
- Momentary push button
  This is a deadman button connected to the motor driver resets through a buffer and to the microcontroller.

High power components:

- Motor driver (2): Pololu High-Power Motor Driver 34V/23A
  Each motor driver is rated to 23A continuous current without a heatsink. We have mounted heatsinks on the motor drivers to increase the continuous current to above 30A.
- Motor (2): MY1020Z 24VDC, 500W
  Each of the motors can has a max torque of 12.6 Nm which is 84.4Nm with a 6.7:1 reduction gearing (no mechanical loss). The rated speed is 375rpm. Max current is 26.7A.
- Batteries (3): 3000mAh 6s 20C LiPo pack
  Each battery is capable of delivering 60A. This circuit uses three of these batteries in parallel. The voltage of the battery is 21.6V to 25.2V depending on the power left in the battery.

Fig. 3 – Code for the Robot The figure below shows a flowchart of the code running on the robot. The code for the robot is written in C/C++, using some libraries and functions from the open source platform Arduino. The code consists of a part that initializes the system followed by an infinite loop that balances the robot and sends data to the Android application via Bluetooth. **Initialization:** When the robot is started up, values are read from the EEPROM, outputs and inputs are initialized on the microcontroller and the steering is calibrated. **Main loop:** The main loop is checking for IMU-data, updating PID-values, measuring battery voltage and sending serial data via the Bluetooth module to the Android application.





BalancingRobotFullSizeAndroid
by Mads Bornebusch and Kristian Lauszus
Current draw: 0.00A
Turning value: -0.19
Battery level: 26.03V
Run time: 0 min 36 sec

Current Values:
P: 2.80 I: 2.00 D: 3.50
Target Angle: 0.00 Turning scale: 40
Adjust Values:
P: 2.80
I: 2.00
D: 3.50
Target Angle: 0.00
Turning scale: 40

✓ Accelerometer Q_{angle} 0.0010
✓ Gyro Q_{bias} 0.0030
✓ Kalman filter $R_{measure}$ 0.0300

Fig. 4 – Andriod Application The figure shows screenshots of the four different screens in the Android application written for the robot. **Top left:** This screen shows different values that is sent from the robot to the app. This is general information for the user such as current draw, battery level and PWM-value. **Top right:** This screen is for adjusting the PID-values for the controller. This is used for tuning the robot to make it balance. The user can also set the target angle and how aggressively the robot is turning. **Bottom left:** This screen is showing the user a map with the current position of the robot using the Android device's built in GPS. This can be used for navigation and to see which attractions are nearby. **Bottom right:** This screen is showing a graph with the angle taken from the gyroscope, the accelerometer and from the Kalman filter.

Protocol: In order to make sure that the data is parsed properly between the robot and the Android application, a protocol for this was implemented. It consists of a constant string header, one byte to indicate the command and one byte indicating the length of the data to follow. After the data there is a checksum calculated by taking XOR of all bytes in the message excluding the header.