Special Vehicle for Transporting Unstable Chemicals
Design Review

Team 15
Zhangxiaowen (Andy) Gong
Wenjia (Airie) Zhou
Jun Ma
TA: Justine Fortier

University of Illinois at Urbana-Champaign

March 6, 2012

Contents

1 Introduction

1.1 Motivation

Our team would like to implement a special vehicle (robot) for transporting unstable chemicals. As engineering students who are required to take chemistry courses and perform experiments, we clearly know that safety always has top priority when dealing with chemicals. Besides, proper transportation of volatile substance is also essential for experiments to yield correct results. However, although people treat chemicals with all the care, hazard still exists, maybe in moving a flask from one desk to another in a crowded lab room, or maybe in bringing a freezer from Chemistry Annex to Noyes Lab through the underground tunnel. As a result, we would like to create a transportation tool that fits both compact environment (e.g. lab room) and road transportation. In addition, the project can be extended to other uses such as lunar rover, wheel chair, etc. We are looking forward to see this special vehicle shuttles between Noyes Lab and Chemistry Annex in the future.

1.2 Objective

The major goal for the project is to keep the chassis of the vehicle stable. To reach that, the mechanical system adopts a novel movement scheme that is able to keep the vibration of chassis in a tolerable range with the help of a control system. In order to facilitate both the calibrating process during the development stage and the performance evaluation afterwards, the system also equips a real-time data acquisition system that transmits sensor readings to a computer for further analysis. So far, the system intends to provide the following features:

1. a four-wheel omni-direction movement scheme
2. a symmetric design that allows any of the four sides of the vehicle to dynamically take over as the forward direction
3. an active suspension system on each wheel
4. a control system that stables the chassis based on the feedback from MEMS sensors
5. a wireless link used for transmitting real-time sensor data to a computer for performance analysis

The project also offers the following benefits:

1. enhancing safety level in road transportation for chemicals
2. easing people's tension when moving dangerous chemicals in the lab

2 Design

2.1 Block Diagram

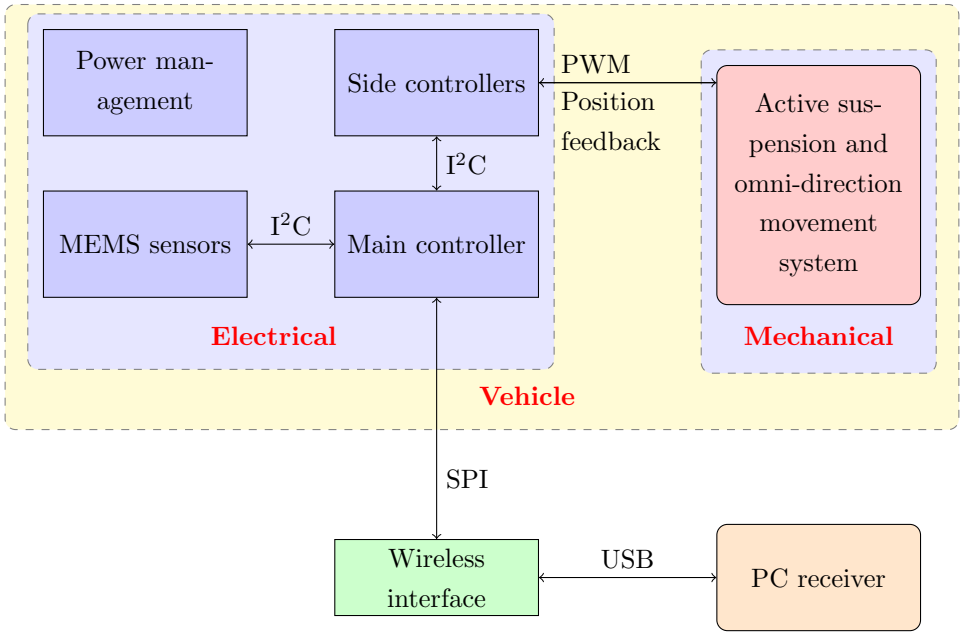


Figure 1: Block diagram

2.2 Block Descriptions

2.2.1 Electrical/Electronic Part

The electrical and electronic portion consists of six modules:

1. **Main Controller:** Since the system performs various functions ranging from filtering sensor data to controlling motors, we adopt multiple microcontroller. The main controller is based on a 32-bit AVR microcontroller that performs the following tasks:

   (a) gathering data from the MEMS sensors and using filter to reduce noise

   (b) making decision to balance the chassis according the the processed sensor data

   (c) printing debug information on an OLED (Organic Light Emitting Diode) display

   (d) sending real-time sensor data to a PC via wireless

   When an adjustment to the chassis is required, the main controller commands side controllers to drive 12 actuators in total. The software for the main MCU is built upon an RTOS (Real-Time Operating System).

2. **Side Controllers:** The side controllers is based on 8-bit AVR microcontrollers. Each MCU receives commands from the main controller and controls 1 servo, 1 regular DC motor, and 1 worm motor. They complete the following tasks:

   (a) driving motors and servos

   (b) operating the active suspension system in a closed-loop manner

   (c) adjusting four wheels to actualize omni-directional movement

   Furthermore, each side controller has one H-bridge that drives the active suspension. In order to achieve minimal dissipation as well as optimal driving capability, we will not use integrated H-bridges; instead, we decide to implement the circuits using discrete components. The software for the side controllers is built in a foreground-background manner.

3. **MEMS sensors:** Since we use accelerometer, gyroscope, and magnetometer to measure the state of the chassis, the InvenSense MPU-6050 becomes cost-effective since it integrates both accelerometer and gyroscope. moreover, it can acquire data from a third-party 3-axis magnetometer via a built-in I^2C (Inter-Integrated Circuit) master controller so that it can perform 9-axis sensor fusion inside the chip. We choose HMC5883L, a 12-bit 3-axis magnetometer, to act as MPU-6050's auxiliary sensor. The main controller get data from MPU-6050 through I^2C protocol. Since MPU-6050's I^2C master controller has limited functions, it needs the main controller to configure the auxiliary sensor before it can read the output. To resolve the issue, MPU-6050 has an internal MUX that can temporarily hand the auxiliary sensor over to the main controller for initialization.[4]

4. **Power Management:** The vehicle is powered by a 3-cell 11.1V 3000mAh LiPo battery. The on-board step-down and step-up regulators then provide 3.3V, 5V, and 12V power for different uses. The power management scheme for the model vehicle is designed to support the whole system for 1 hour of continuous operation on heavy load, which is enough for demo purpose.

5. **Wireless Interface:** In order to analyze the system performance, the vehicle constantly send the sensor readings wirelessly to a host computer. We choose a nRF24L01+ 2.4GHz RF solution to fulfill the task. The MCUs can send commands and packages to the RF chip via SPI (Serial Peripheral Interface).

6. **PC Receiver:** An 8-bit AVR microcontroller that has a built-in USB controller handles the computer side RF chip, reconstructs the received data into USB (Universal Serial Bus) packages, and sends them to the host computer. In addition, a computer software will be developed to visualize the data, to store/playback the data, and to calculate the statistics for further analysis.

2.2.2 Mechanical Part

The main mechanical challenges in the project are the active suspension system and the omni-directional movement system.

1. **Active suspension:** Although hydraulic actuators are preferable for the final product, the reduced-size model system uses worm motors to operate the suspension. In order to monitor the current position of each suspension, a potentiometer will be attached to the motor and gives analog feedback that can be read by the controller's ADC (Analog to Digital Converter). In order to build a PID control system over the worm motor and its feedback, we have to calculate the motor's transfer function. The worm motor consists of a DC motor and a worm drive system. The transfer function of the DC motor is $G_m = \frac{K_m}{(s\tau_m+1)}$[2] We will use MATLAB to calculate the constants in the equation. The transfer function of the worm drive is $\frac{1}{i}$, where i refers to the gear ratio. Therefore, the transfer function of the whole motor is the product of the two transfer functions described above. The worm motors are driven by the H-bridges described above.

2. **Omni-directional movement:** Since the typical omni-wheels (http://en.wikipedia.org/wiki/Omni_wheel) that are available for purchase vibrate when rolling, we consider to use a servo to adjust the z-axis angle of regular wheels instead. The controller can change the positions of the servos by sending them PPM (Pulse Position Modulation) signals. By designing the whole vehicle symmetric respecting to both x-axis and y-axis, the vehicle can adopt a new forward direction and shifts when it is supposed to turn. Applying this scheme means that a separate motor is required to drive each wheel, which introduces an issue that different motors may output varied speed. To solve that, the controller can adjust the absolute speed of each motor by reading the encoders mounted on the wheels.

2.3 Schematics

2.3.1 Main Controller

2.3.1.1 Microcontroller The main controller uses AT32UC3C2256 as processor. The communication modules that are utilized in the design include I^2C (a.k.a TWI, Two-Wire Interface), SPI, and USB. We also connect the external interrupt pins of the MCU to other modules. Figure 2 is the core circuitry that supports the MCU, including oscillator, power filtering circuit, USB interface for bootloader, reset circuit, etc.

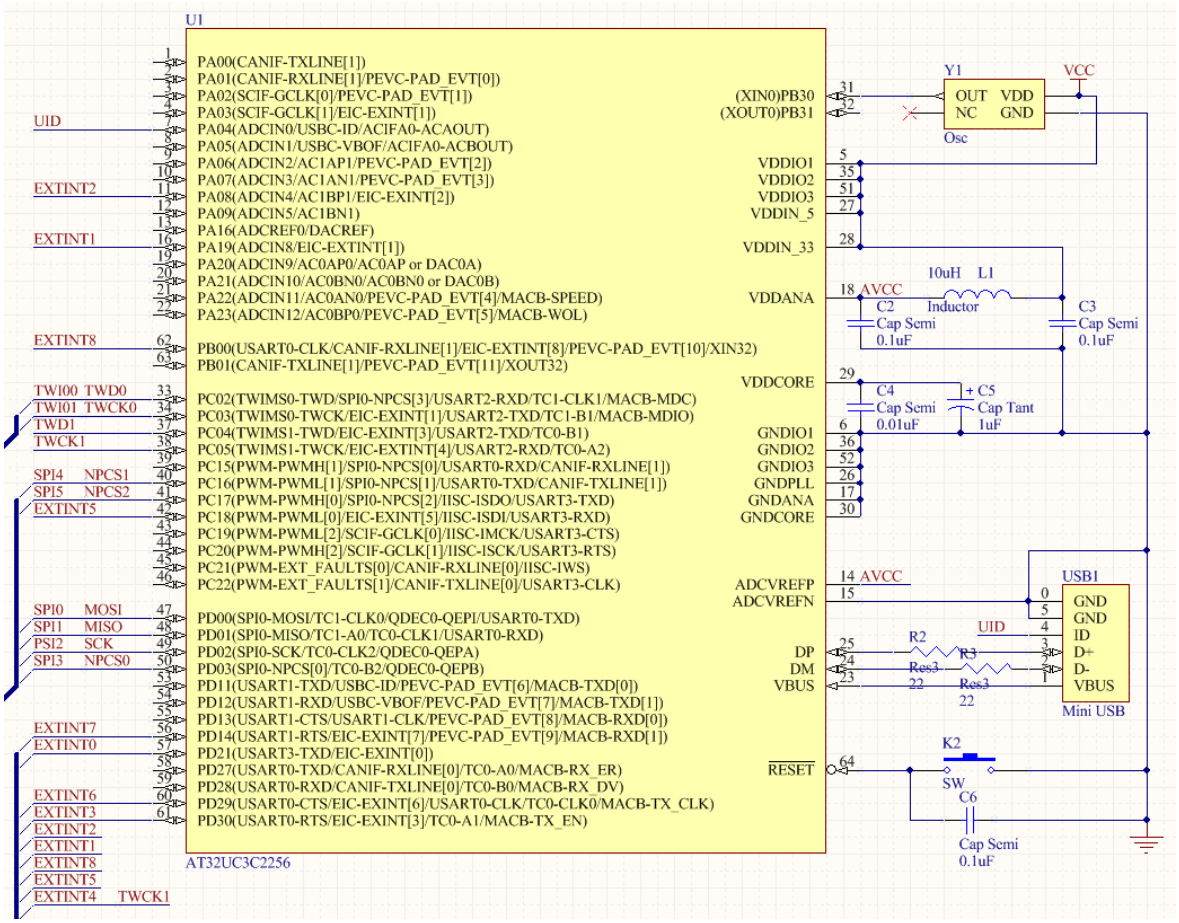


Figure 2: AT32UC3C2256 core system

Since the main MCU runs at 3.3V while the side ones work at 5V, a bidirectional level converter is required to establish the I^2C bus between them. We adopt a design that uses N-channel MOSFETs in a symmetric manner[6], which is presented in Figure 3. In the figure, VCC is 3.3V. Label TWD1 and TWCK1 belong to the 3.3V bus, and label TWI10 and TWI11 are the 5V ones. The circuit in the figure looks not symmetric; in fact, the 3.3V lines do have pull-up resistors although they are not shown in the figure due to being inside a resistor pack.

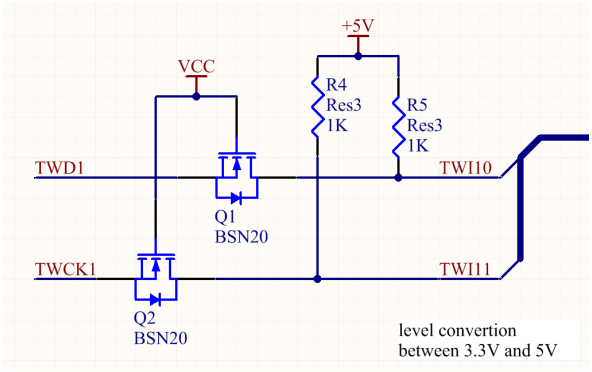


Figure 3: Bidirectional level shifter

2.3.1.2 Power Management The power management part on the main controller provides 3 distinct positive voltage levels: 5V, 3.3V, and 12V. Because we use a 3-cell LiPo battery to power up the system, we embrace a special voltage regulating scheme that first regulates the approximately 11.1V supply voltage to 5V and then step it up to 12V as well as down to 3.3V. The 5V regulator we use is LM2596, a highly efficient switching converter that works at 150KHz and has 4A current output capacity. It supplies power for both generating the other two voltages and serving the side MCUs. Figure 4 includes its work state switch and output filtering circuit.

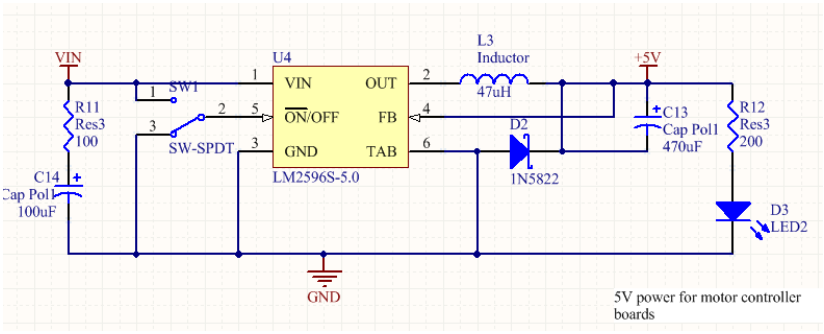


Figure 4: 5V regulator

We use a MAX629 switching step-up regulator for 12V supply. The chip needs two resistors to designate the output voltage from its internal 1.25V reference, $R_1 = R_2 \times (\frac{V_{out}}{V_{ref}} - 1)$[5] Since the voltage is for biasing the OLED screen and providing the V_{GS} to the MOSFETs in the H-bridge, its error tolerance can be as big as 10%. As a result, we pick two widely used resistance value: 43KΩ as R_1 and 4.7KΩ as R_2 in the formula, which results in the output voltage being 12.6V. Figure 5 shows the circuit.

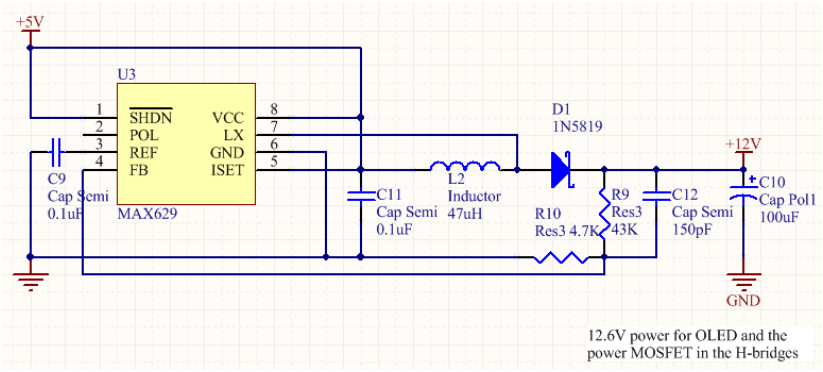


Figure 5: 12V regulator

Instead of using a switching regulator, we use LM1117, a linear regulator with 500mA maximum output current, for stepping 5V down to 3.3V. The reasons for this choice are:

1. The 3.3V system consists of the 32-bit MCU, 2.4GHz wireless transceiver, MEMS sensor, and the logic supply for OLED display, which draw a small total current.

2. The voltage drop from 5V to 3.3V is only 1.7V. Suppose the average current consumption by the 3.3V system is 100mA, the average dissipation power caused by the voltage drop is $1.7V \times 100mA = 170mW$, which is acceptable.

3. The linear regulator requires much less external components that a switching one does.

Figure 6 presents the circuit.

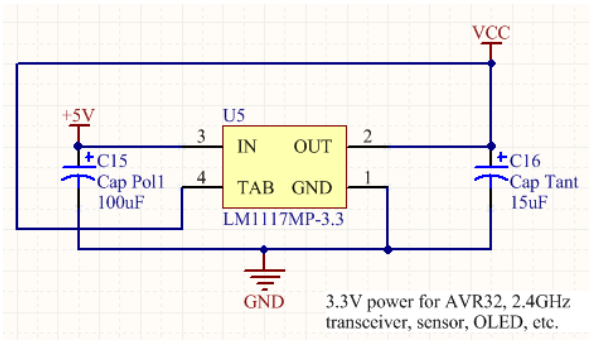


Figure 6: 3.3V regulator

2.3.1.3 MEMS Sensors In order to perform the 9-axis sensor fusion within the MPU-6050 chip, the HMC5883L sensor is connected as an auxiliary as shown in Figure 7. The MPU-6050 further communicates with the main MCU via its I^2C slave interface.

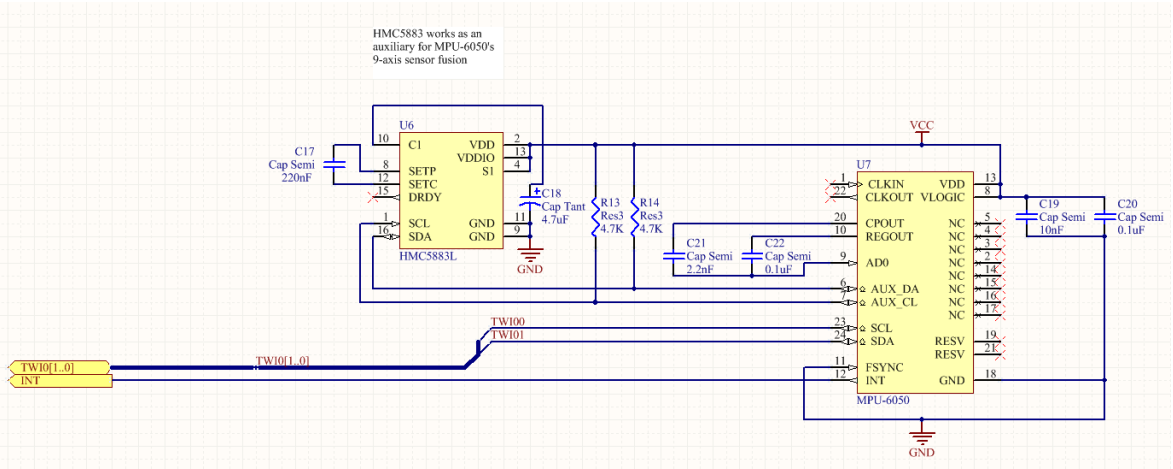


Figure 7: MEMS sensors

2.3.1.4 OLED Display The OLED display takes two voltage supplies: one 12V for biasing the display and one 3.3V for powering the logics. It takes data from the main MCU's SPI interface. Figure 8 shows the circuit.

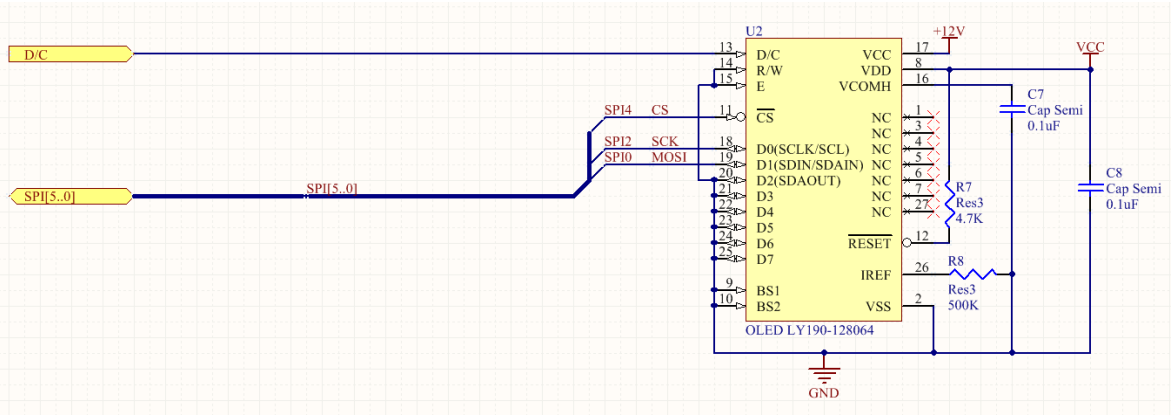


Figure 8: OLED display

2.3.1.5 Wireless Connector Since we are using a module integrating the nRF24L01+ with power amplifier and low noise amplifier, the interface with the wireless module is just a header, which is shown in Figure 9.

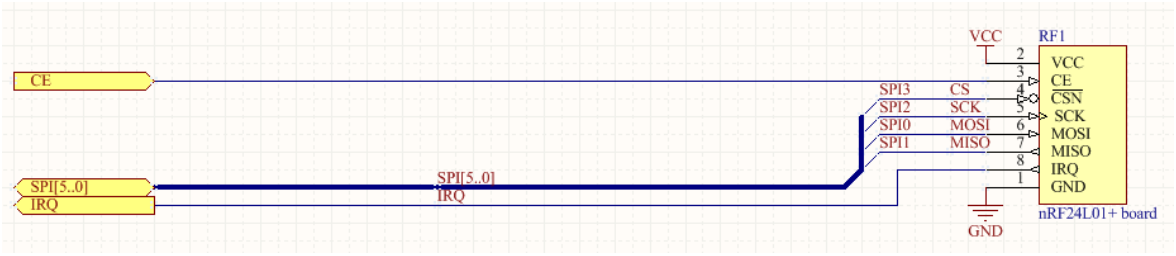


Figure 9: Header to 2.4GHz wireless module

2.3.2 Side Controller

2.3.2.1 Microcontroller The side controllers use ATmega8A as processors. Besides GPIO (general purpose input/output), we utilize the MCU's PWM (Pulse-Width Modulation) generator, external interrupt, and I^2C interface. Figure 10 is the core system, including oscillator, power filtering circuit, reset circuit, etc.

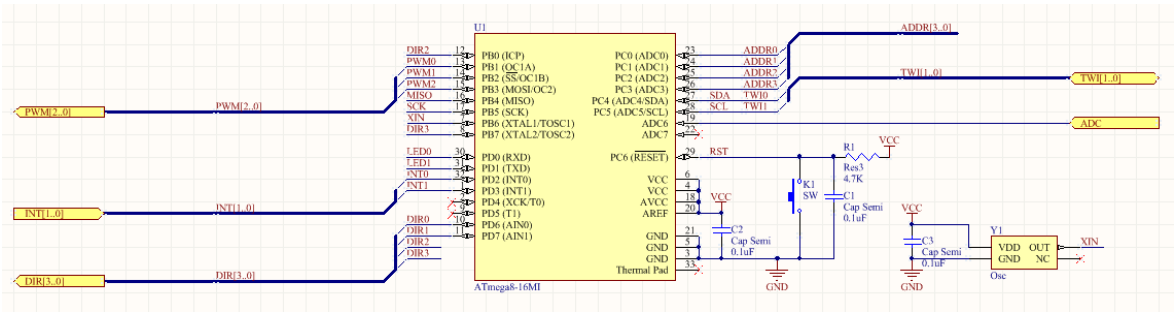


Figure 10: ATmega8A core system

Because we have totally 4 identical side controllers in the system, in order to apply consistent programs to every chip, we connect a 4-bit DIP switch, which is shown in Figure 11, to the MCU's GPIO so that each chip can determine the low 4-bit of its 7-bit I^2C slave address itself.

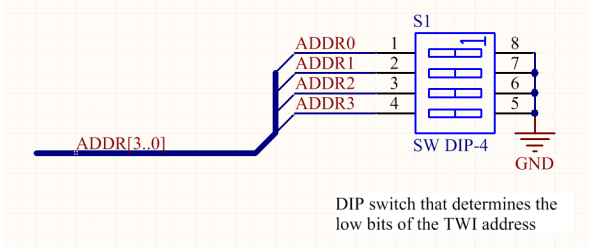


Figure 11: DIP switch for changing I^2C slave address

The microcontroller is powered up by the 5V that comes from the main controller. However, each side controller still embeds a LM2596 regulator. The extra 3A current capability provided by the regulator goes to the 5V DC motor that drives the wheel.

2.3.2.2 H-bridge Figure 12 is the design of the H-bridge we use to drive the worm motors. In the circuit, we use two IR2104 half-bridge chips to provide a 520 nanosecond deadtime to prevent instant short circuit when switching the direction of the motor being driven. Besides, we also have other circuitry consists of Schottky diodes and resistors that protects the chips from reverse EMF (electromotive force). Note that IR2104 requires N-channel MOSFET instead of typical P-channel ones for the upper arms in the H-bridge. In order to provide the floating V_{GS} for the NMOS in the upper arms, a bootstrap circuit built from a diode, a Tantalum capacitor, and a charging resistor, is demanded. In order to charge the capacitor in the bootstrap circuit, we need a certain time of low level in the driving signal. As a result, the PWM signal provided to the H-bridge cannot reach 100% duty cycle; instead, the peak duty cycle is limited at about 95%. The logics on the lower left side of the figure is responsible to translate the direction and PWM signals into logics that IR2104 accepts. Thanks to the logic, we are able to generate forward, backward, and breaking commands from two direction inputs

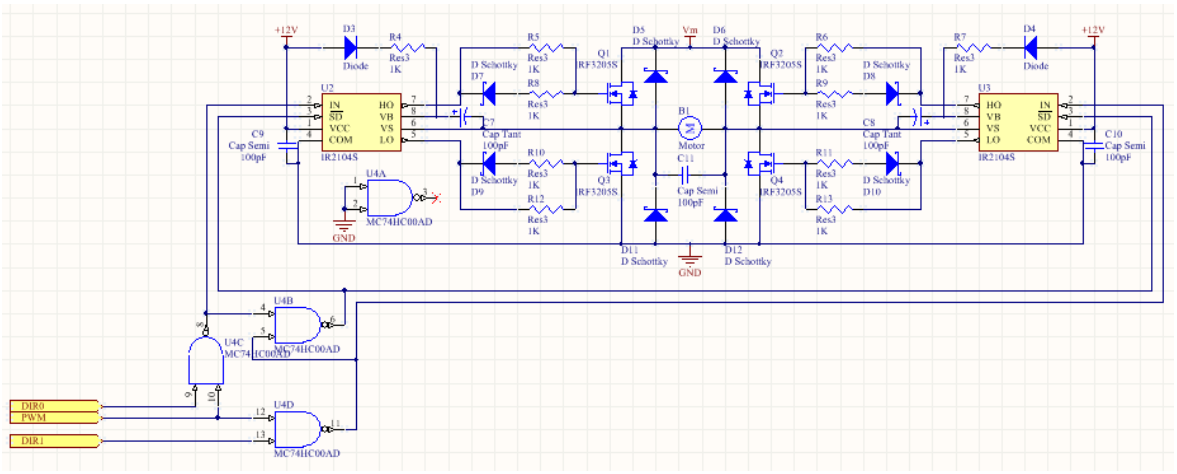


Figure 12: H-bridge

2.3.3 PC Receiver

The MCU used in on the PC side receiver is ATmega32U4. We use two communication modules provided by the chip: USB for PC communication and SPI for interfacing with the wireless chip. Figure 13 is the core system of the MCU. Its USB interface is also utilized by its built-in bootloader. The wireless module on the receiver board is identical to that on the main controller. The receiver board also has an LM1117 regulator for powering the wireless module up.

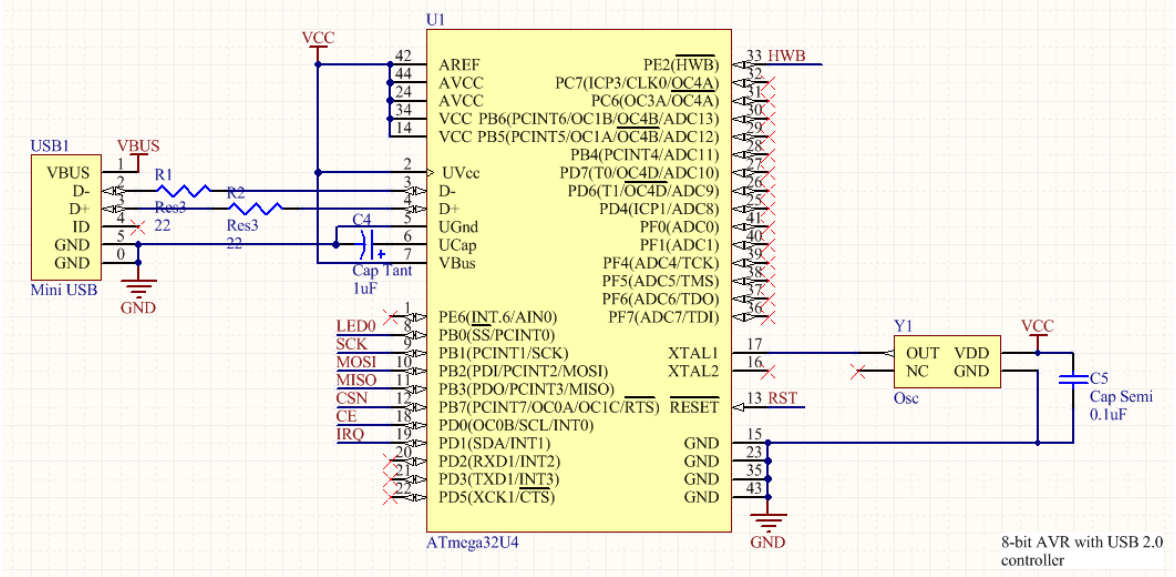


Figure 13: ATmega32U4 core system

2.4 Software Design

2.4.1 General Software Flow

The software for the main controller is based on real-time operating system, which is multi-tasking. As a result, a single flowchart is insufficient for stating the cooperative relationship among tasks. Nevertheless, the following chart gives a general view on the software logic.

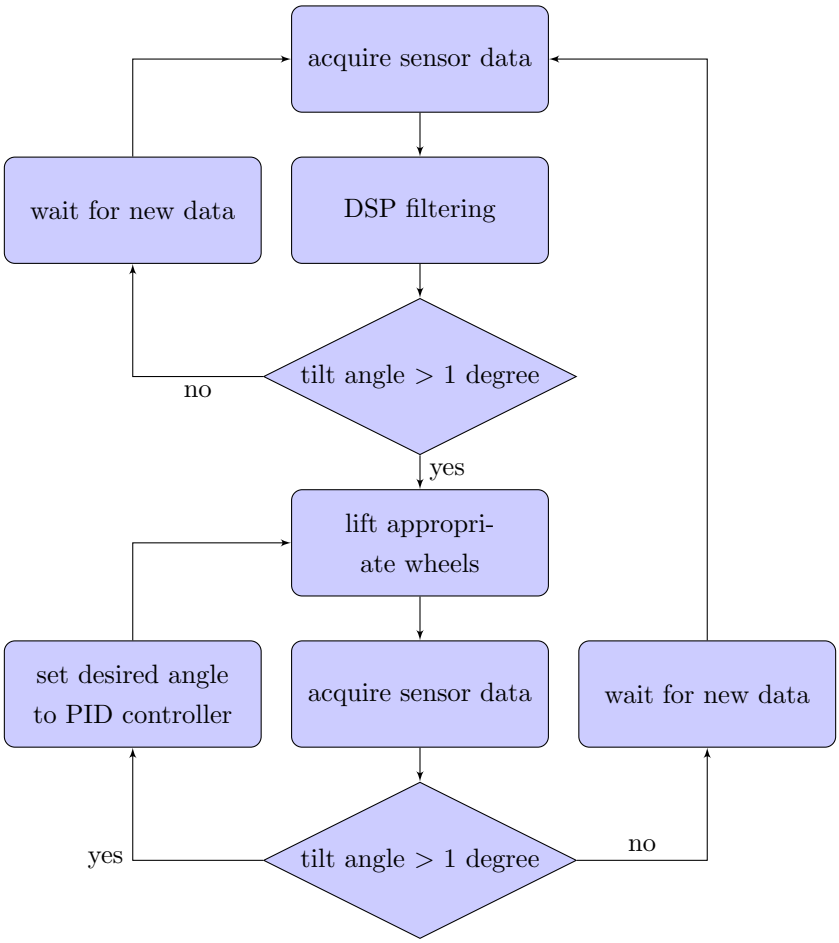


Figure 14: General software flowchart

2.4.2 SPI and I^{2}C Drivers

The major communication interfaces that we utilize on the main controller are SPI and I^{2}C. Since SPI and I^{2}C work at about 10MHz and 400KHz respectively, which is far slower than AVR32's 66MHz CPU clock, we have to design asynchronous drivers for them in order no to let I/O interfaces slow down the whole system performance. Furthermore, we apply a streaming mechanism on both SPI and I^{2}C drivers for optimizing the performance when the drivers are used by multiple tasks. The scheme is based on a variable length queue of packages. We also need to implement a slab cache for the package data structures in order to optimize the use of dynamic memory.

Since each transfer via SPI interface contains relatively large amount of data (less then 100 bytes), we consider to use the DMA (Direct Memory Access) controller in the AVR32 to facilitate the memory operations. The following chart provides an overview on the structure of the SPI driver. The pink blocks are optional call-back functions that allow the task that uses the driver to wrap a package with preface and clean-up routines.

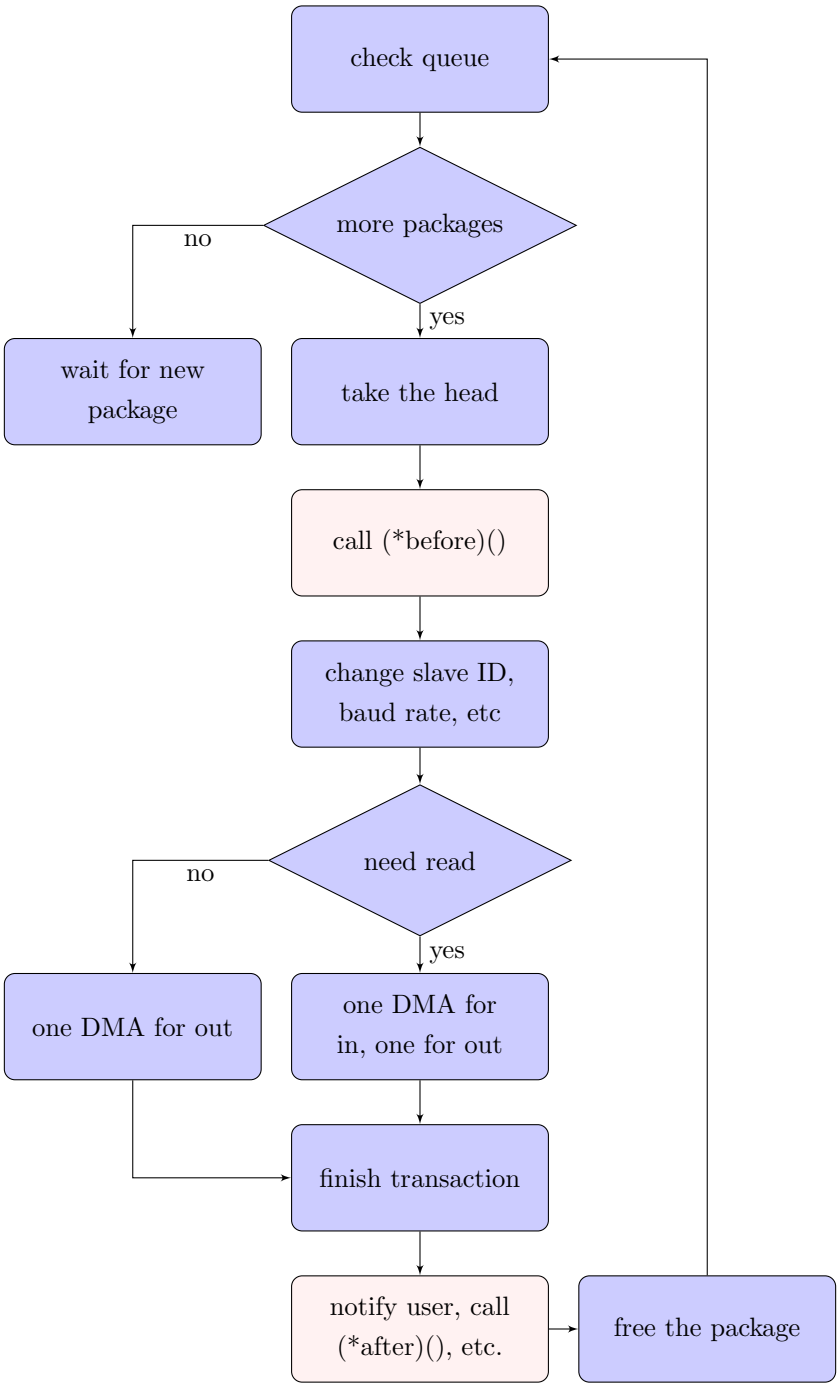


Figure 15: SPI driver

The I^2C driver is much alike. However, since its packages are relatively small (around 10 bytes), currently we do not consider using DMA for it. Figure 16 is its flow chart.

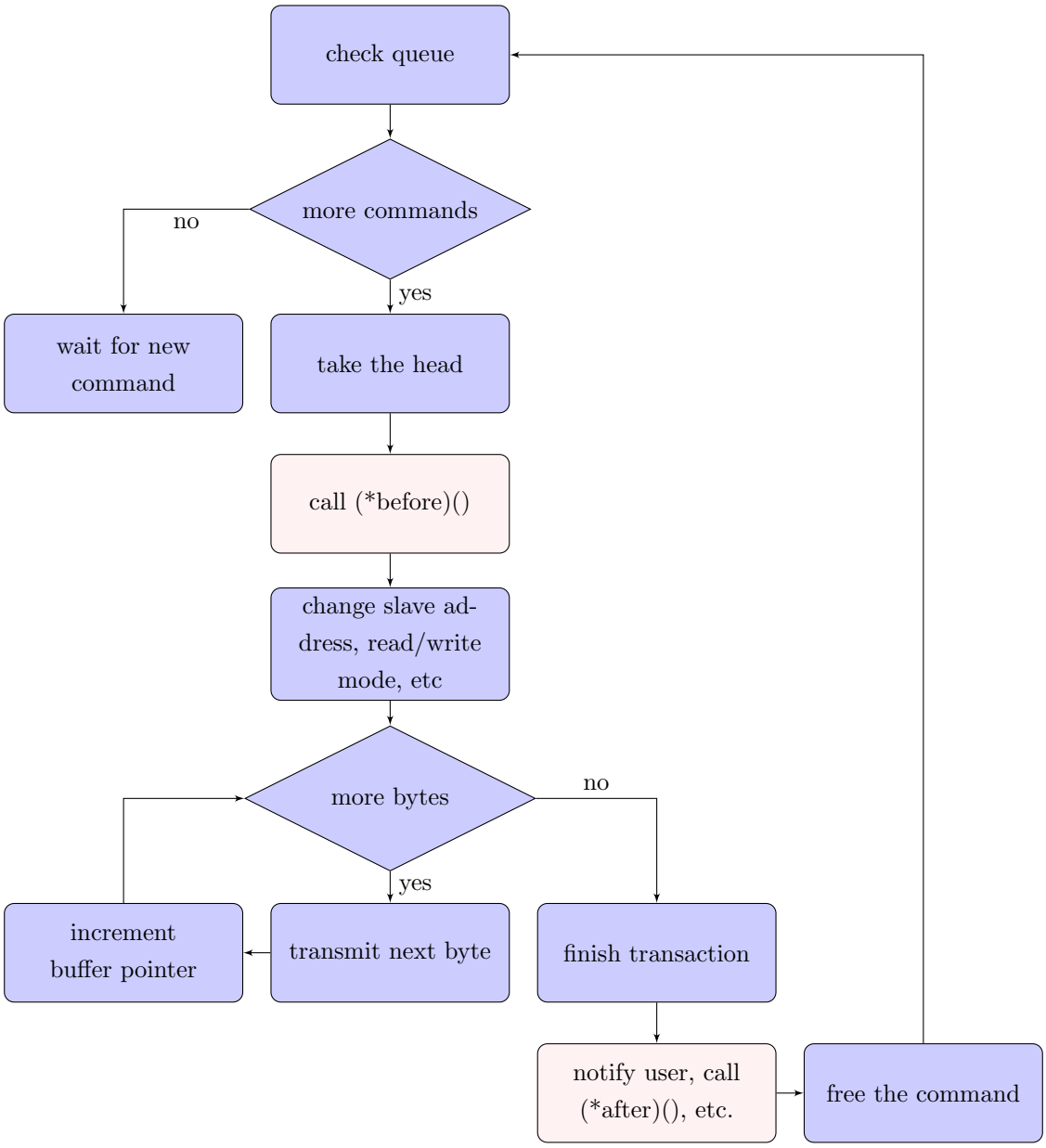


Figure 16: I^2C driver

2.4.3 Wireless Interface

The wireless transceiver that we introduce to the project is nRF24L01+, which works on 2.4GHz∼2.5GHz frequency range and has up to 2Mbps on air data rate. It also uses a built-in protocol called Enhanced ShockBurst^{TM} to perform handshaking and error checking so that the MCU can save computing power for other tasks. After subtracting the expenses on the protocol and the existing sensor data transmission, we still got about 1.5Mbps bandwidth for transmitting packages, which is sufficient for use in

this project.

The driver for the transceiver embraces the same design idea with the lower level peripheral drivers that we described previously. The driver takes buffers from user and streams them out in a FIFO manner. Since the Enhanced ShckBurst^{TM} protocol allows variable length package with maximum length at 32 bytes, if the buffer given by a user has content larger than 32 bytes, the API for appending package will first recurse to divide the buffer into multiple packages and then add them into the send queue. The transceiver has two work mode: PTX (primary transmitter) and PRX (primary receiver), our strategy is to configure it as PRX when it is idle, and only switch it to PTX when the send queue has packages waiting to be streamed out. The transceiver has 3 maskable interrupt source, and uses a pulse on the IRQ pin to report an interrupt. The sources are: RX_DR (RX data ready), TX_DS (TX data sent), and MAX_RT (TX maximum retry reached). The driver makes use of all of them, and to handle them as follow:

- RX_DR:
    1. Read the length of the incoming package
    2. Allocate a buffer that fits the length
    3. If the receive queue has too many packages waiting for the user to fetch, then remove and free the head from the receive queue
    4. Add the incoming package to the tail of the receive queue
- TX_DS:
    1. Check if more packages are in the send queue. If so, continue to send the next package
    2. If not, go to PRX mode
- MAX_RT, this interrupt is triggered when the transmitter loses connection with the receiver, so we handled it as:
    1. Flush the TX FIFO in the transceiver
    2. Rewrite the last package into the TX payload

In order to pair with the main controller, the receiver on the PC side also uses an nRF24L01+. Although the driver for it is also based on stream, it handles the modes differently. Unlike the driver on the AVR32, which switches to PTX for sending packages, the driver on the receiver side always stays at PRX. When there are packages ready to send, it utilizes the acknowledgment package specified in the Enhanced ShckBurst^{TM} protocol to send them. An acknowledgment package is the same with a normal TX package in term of length; however, it is sent passively when the receiver is acknowledging the transmitter. Therefore, it is never sent until the link is established between the transmitter and the receiver; thus, the MAX_RT interrupt will never happen under this configuration.

2.4.4 USB Receiver

We use USB interface to send wirelessly received data to the PC. The MCU chosen for the job is ATmega32U4, which is an 8-bit AVR with built-in full-speed USB 2.0 interface, 32KB FLASH program space, and 2.5KB SRAM data space. In order to facilitate the development for the PC side driver, we decided to configure the receiver as a CDC (Communication Device Class) device, which is natively support by Microsoft Windows and other operating systems. A CDC device essentially emulates a serial port. For example, the widely used FTDI chips are all CDC devices. However, since our driver does not actually bridge the USB packages to a physical UART, the data rate will not be limited by the UART. In addition, since we intentionally ignored the SET_LINE_CODING command from the PC, which is specified in the CDC to set up the baud rate, parity bit, stop bit, etc. of the device, we could use any baud rate to open the virtual serial port on the PC and still receive correct package at high data rate.

The USB controller in ATmega32U4 has a control endpoint (EP0) with a bank up to 64 byte and 6 other programmable endpoints. Besides the default EP0, the CDC specification uses other 3 endpoints. One is used by the CDC class interface, and the other two are used by the data interface. We configure the endpoints in the AVR as follow:

- Endpoint 1 is configured in bulk IN mode and has two 64 byte banks in ping-pong mode. It sends the data from the wireless interface to the PC.
- Endpoint 2 is configured in bulk OUT mode and has a 64 byte single bank. It receives data from the PC.
- Endpoint 3 is configured in interrupt IN mode and has a 64 byte single bank. It works as the management endpoint for the CDC class interface. This endpoint is essentially a dummy one. We do not need to handle any event from it.

After finish the embedded software, we also need to install the PC driver to test it. Since Windows has built-in driver for CDC devices, all we need to do is constructing an .inf file that describes the device. In order to identify our device, we have to set up the VID (Vendor ID) and PID (Product ID) first. Nonetheless, applying for valid VID and PID is expensive. Fortunately, ATMEL provides several sets of free VID and PID for products that uses AVR. The free set for CDC devices are VID=0x16C0 and PID=0x05DF.

2.4.5 PC Software

We decide to use QT as the PC software develop environment because it is cross platform and has a powerful GUI library. The final PC software is capable of reading, saving and loading the data received from the wireless interface. It is also able to generate a real time line chart from the received data.

2.4.6 DSP Filter

After we perform tests on the built-in sensor fusion in the MPU-6050 chip, we will evaluate this hardware function and determine whether the data it generates are satisfying. If not, we may need

an additional software filter to eliminate undesired noise from the raw sensor data. At this moment we have two options for the filter, one is the Kalman filter and the other is complimentary filter. Both filters work towards the function we expected; however, differences do exists. Kalman filter's characteristic makes it very sensitive to noise.[3] Complimentary filter, in contrast, is not as sensitive but react faster since the algorithm in the latter one is simpler than that in the former one.[1] Right now we can't decide which filter is better for our design, since both sensitivity and speed of processing are required in signal processing. If we consider that complimentary filter is sensitive enough after testing, it becomes preferable since less computing power is needed. Otherwise, Kalman filter is our only choice. In addition, the controller has to perform all the tasks described in the block description at the same time. Among them, the possible software filter requires the most intense calculation. As a result, the RTOS based program has to balance the computing power for different tasks so that all of them can run smoothly without interfering each other. Figure 17 presents the general difference between the two filters, and Figure 18 shows how the filtered data is used in the later control system.

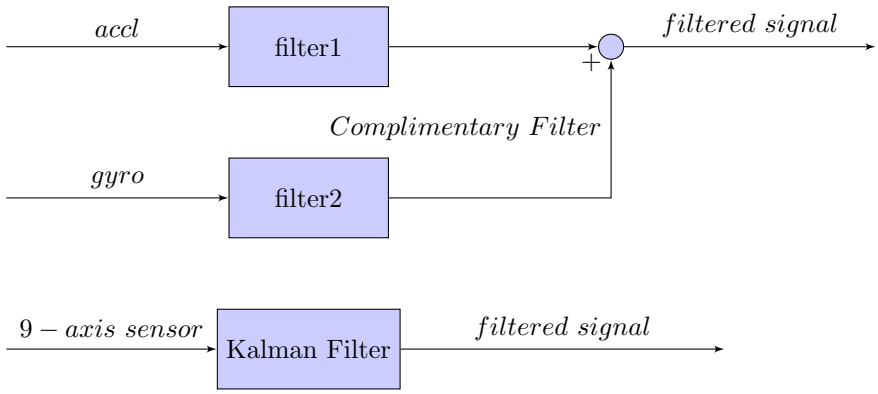


Figure 17: Complementary filter and Kalman filter

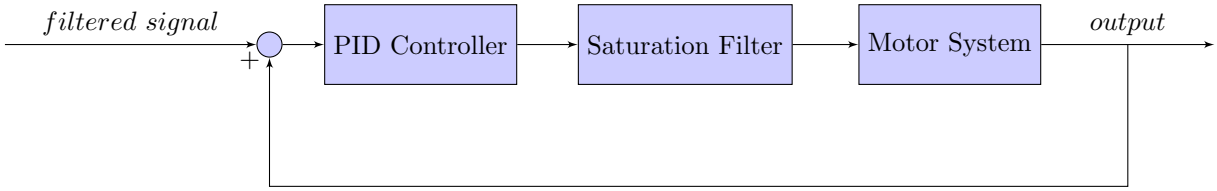


Figure 18: Filtered data as input to the control system

3 Requirement and Verification

3.1 Modulated Requirement and Testing

3.1.1 Main Controller

Table 1: Requirement and verification for the main controller

Item	Requirement	Test/calibration procedure
OLED display	1. Display characters at specific coordinates	1. (a) Command the OLED to display strings with various lengths at different location (b) See if all 128×64 pixels are operational (c) See if all ASCII characters are mapped to pixels correctly (d) See if the designated coordinates are translated to right positions (e) Check if the strings wrap around correctly
DSP filter (if present)	1. Finish each filter computation within 10ms 2. Filter at least 90% vibrational noise out	1. (a) Turn on the built-in 16-bit T/C (timer/counter) before starting the calculation and record the value in the T/C register right after the computation finishes (b) Calculate the delay from the T/C reading and its clock source setting 2. (a) Send both filtered and raw data to a computer (b) Calculate and compare the standard deviation of each set of data

Continued on next page

Table 1 – continued from previous page

Item	Requirement	Test/calibration procedure
I^2C interface	1. Signals have almost vertical edges maximum clock speed at 400KHz 2. No malfunction in high-noise environment 3. Communication between 3.3V I^2C bus and 5V I^2C bus 4. The streaming algorithm is able to save over 70% CPU cycles from simple synchronous drivers	1. (a) Set the I^2C bus to 400KHz and then use an oscilloscope to probe the SCL and SDA lines (b) Check the edges on the signals and determine of the curve on the edges, which are caused by the capacitance on the trace/wire and the pull-up resister specified in the I^2C standard, is in an acceptable shape 2. Test the transmission when the H-bridges and motors are turned on, and see if any data loss occurs 3. Test the level shifting circuit to see if it converts signals bidirectionally 4. Loop to append 100 I^2C transactions and see how many packages remain in the queue after the appending operation is done; the more packages still stay in the queue the better the algorithm is

Continued on next page

Table 1 – continued from previous page

Item	Requirement	Test/calibration procedure
SPI interface	1. Internal shift register is functional 2. No malfunction in high-noise environment 3. When addressing one slave device, no interference coming from the others 4. Utilize the DMA channel for both transmitting and receiving data	1. Check if the slave data shift to the master correctly when the master shift its data out 2. Test the transmission when the H-bridges and motors are turned on, and see if any data loss occurs 3. (a) Pull down the chip select lines on different slaves and see if data go to correct receiver (b) Check if the master changes SPI mode (clock phase, clock polarity, etc.) correctly when switching among slaves that adopt different modes (c) Keep addressing various slaves for 10 minutes and capture any malfunction 4. (a) Turn on the inner-loop function of AVR32's SPI module, which interconnect its input to its output (b) Set one DMA channel to transmit data in a block of memory (c) Set the second DMA channel to receive data and to store them in another block of memory (d) After the transmission finishes, check the consistency of the content in the two memory blocks

3.1.2 MEMS Sensors

Table 2: Requirement and verification for the MEMS sensors

Item	Requirement	Test/calibration procedure
MPU-6050	1. Be configured with proper work mode (sensitive range, interrupt output, etc.) 2. Allow the main controller to configure its auxiliary sensor (HMC5883L) 3. Accelerometer and gyro scope data are consistent	1. (a) Read values from all registers from the sensors' register map and check with the default value stated in their datasheets (b) Change values in several registers and then read back to see if the operations succeed (c) Move/turn the sensor at certain rate and check if the output is in the expected range 2. (a) Use the sensor's internal MUX to hand the auxiliary sensor to the main controller temporarily (b) Let the main controller to verify if it can change the auxiliary sensor's registers correctly (c) Get the auxiliary sensor back from the main controller and try to read its output (d) Let the main controller to address the auxiliary sensor again; the operation should fail now 3. (a) Put sensor unit on the table, tilt a little bit, and record the output (b) Calculate θ from accelerometer's x-axis (x) and z-axis (z) data: $\theta = \arctan(-\frac{x}{z})$ (c) Calculate γ from gyroscope's z-axis (ω) output: $\gamma = \omega \times t$ (d) θ should be close to γ

Continued on next page

Table 2 – continued from previous page

Item	Requirement	Test/calibration procedure
HMC5883L	1. Be configured with proper work mode (gain, interrupt output, etc.) 2. Error is controlled within 2°	1. Perform similar procedure that checks the MPU-6050 2. (a) Place the sensor on the table and tilt 45° at a time (b) Record the output and compare it to the actual angle; $error = actual - output$ (c) Repeat the step (a) and (b) until the total angle tilted reaches 720° (d) Calculate the peak and average of the errors

3.1.3 Side Controllers

Table 3: Requirement and verification for the side controllers

Item	Requirement	Test/calibration procedure
I^2C interface	1. Robust communication with no data loss and bus errors in high-noise environment 2. Multiple side MCUs can communicate with the main controller via a single I^2C interface without suffering bus conflict	1. Perform the test 1 and 2 in the I^2C part in section 3.1.1 2. (a) Send commands to different slave MCUs to verify the addressability (b) Let the main controller keep changing the slave MCU for 10 minutes and see if the bus fails in the duration

Continued on next page

Table 3 – continued from previous page

Item	Requirement	Test/calibration procedure
PWM	1. Generate 3 PWM signals with different duty cycles simultaneously	1. (a) Write a program that generate 3 PWM signals with different duty cycles and changes the duty cycle every 10 seconds (b) Use 2 oscilloscopes to probe the 3 PWM output pins (c) Check the correctness of each signal over time
H-bridge	1. Require no heat sink when driving current stables at the worm motor's rated current, which is 1.10A 2. Respond to PWM signals that have duty cycles ranging from 15% to 95%	1. (a) Connect an ammeter between the H-bridge output and the worm motor (b) Set the H-bridge to output at full speed (95% duty cycle PWM) and increase the load on the worm motor until the ammeter reading reaches 1.10A (c) Keep the motor spinning for 2 minutes (d) Put on antistatic wrist strap and feel the temperature on the driving MOSFET with bare finger; the finger should not feel hot 2. (a) Set the PWM signal to 15% (b) Let the H-bridge drive a motor with no load for 1 minute and see if the speed is constant (c) Repeat (a) and (b) with a 10% increase in duty cycle each time until the duty cycle reaches 95%

3.1.4 Wireless Communication

Table 4: Requirement and verification for the wireless communication

Item	Requirement	Test/calibration procedure
nRF24L01+	1. Be configured with proper work mode (channel, address, data rate, etc.) 2. Communication range is higher than 100m outdoor 3. Communication is not affected if the transceivers have a wall in between	1. Perform the test 1 in the MPU-6050 part in section 3.1.2 2. (a) Put the transmitter and receiver at the two corners of the Quad (b) Turn on both transceivers and check for stable transmission (c) If fail to establish the data link, hold the transmitter and walk towards the receiver until stable data appears (d) Mark the final locations of the transceivers on a scaled map and measure the distance 3. (a) Put the two transceivers in two adjacent rooms and start the transmission (b) If the data is stable, move the transceivers to rooms that have two walls in between (c) Increment the number of walls in between and perform the test until either the transmission is blocked of the number reaches 5 (d) Repeat the test with transceivers placed in different floors

Continued on next page

Table 4 – continued from previous page

Item	Requirement	Test/calibration procedure
Driver	1. Connect/disconnect automatically when the paired transceivers are in-range/out-of-range 2. Automatically resend packages that is not acknowledged by the receiver 3. The streaming algorithm is able to save over 70% CPU cycles from simple synchronous drivers	1. (a) Make the transmitter keep sending packages (b) Switch the on/off state of the receiver several times (c) See if the link is reestablished automatically 2. In the above test, check if any package is lost during the off state of the receiver 3. Perform the test 3 in the I^2C part in section 3.1.1

3.1.5 Omni-directional Movement

Table 5: Requirement and verification for the omni-directional movement

Item	Requirement	Test/calibration procedure
DC motor	1. With speed control algorithm, all motor respond the speed command ($0 \sim 255$) identically	1. (a) Set the speed commands for all motors to 50 (b) Use a tachometer to record the speed of each motor (c) Repeat (a) and (b) with a 50 increase in speed command each time until the command reaches 250

Continued on next page

Table 5 – continued from previous page

Item	Requirement	Test/calibration procedure
Servo	1. All servos respond to position commands identically, which means turning to the same angle 2. Turn 60° with regular load within 0.25 second	1. (a) Set PPM signals equal to the raw position commands (b) Set the duty cycle of the raw command to 2.5% (0.5 millisecond high level) and provide it to all servos (c) Record the position of all servos (d) Repeat (b) and (c) with a 1.0% increase in duty cycle each time until the duty cycle reaches 12.5% (2.5 millisecond high level) (e) Pick one servo as standard, and then calculate scalar and offset errors of the other servos according to the results (f) Apply proportional and offset factors to the raw command to generate modified signals and do (b) to (d) several times until all servos respond to the raw command identically 2. (a) Put the vehicle on the table and let one servo turn 60° (b) Use a video camera with 60fps frame rate to capture the movement (c) Playback the video frame by frame and count the number of frames taken to finish the turning (d) Calculate the actual delay in second by $delay = \frac{frametaken}{60}$

3.1.6 Active Suspension

Table 6: Requirement and verification for the active suspension

Item	Requirement	Test/calibration procedure
PID algorithm	1. Settling time is controlled in 50ms 2. Error between desired angle and actual turned angle is smaller than 2°	1. (a) Run Simulink in MATLAB and set the desired angle x as input to the system (b) Plot step response in MATLAB and find the time when the output reaches 95% of steady state value (c) Load PID parameter into the vehicle and run it over a ramp with $slope = x$ (d) Collect potentiometer reading from the controller's ADC (analog to digital converter), which indicates the angle turned, and import them to MATLAB (e) Mark the delay between the initial position and the final position by analyzing the data 2. (a) Set 10° as the target angle to the PID system (b) Measure the actual turned angle via a protractor after the system reaches stable state (c) Record the error (d) Repeat from (a) to (c) with a 10° increase in target angle each time until the target angle reaches 50°

3.1.7 PC Receiver

Table 7: Requirement and verification for the PC receiver

Item	Requirement	Test/calibration procedure
USB interface	1. Both PC side driver and MCU side driver are functional 2. Data rate is independent from the virtual serial port's baud rate setting	1. (a) Connect the receiver to the PC's USB port and see if the PC recognize it with the name specified in the .inf file we write (b) Open the corresponding virtual serial port in the HyperTerminal (c) Exchange a group of data and verify the correctness 2. (a) Set the baud rate of the virtual serial port to 240bps (b) Loop to send 1KB of random data to the PC (c) Check whether the transmission completes instantly instead of taking around 4 seconds
PC software	1. Receive real-time data and reflect them on line charts 2. Store data and time in file and playback afterwards	1. (a) Establish the wireless link between the vehicle and the PC (b) Move the vehicle randomly by hand (c) Check if the line charts correctly display the movement without recognizable delay 2. (a) Write received data to a non-existed file and check if the file is created successfully in the file system (b) Open the file and check the consistency with the original data

3.1.8 Power Management

Table 8: Requirement and verification for the power management

Item	Requirement	Test/calibration procedure
Voltage regulation	1. Correctly generate 12V (10% tolerance), 5V (1% tolerance), and 3.3V (1% tolerance) voltages 2. Voltages are stable (5% tolerance) during potential instant high load	1. Use voltmeter to probe all 3 generated voltages 2. (a) Build a voltage divider to translate all three voltages proportionally to 2V range (b) Temporarily connect the three translated voltages to the main controller's ADC input pins (c) Run the vehicle for 5 minutes and send the ADC readings real-timely to the host computer (d) Calculate the standard deviation and the minimum of the data (e) Replace the bypass capacitors with ones having higher capacitance if instant voltage drops are observed, and then repeat from (a) to (d)
Battery	1. Support the vehicle for 1 hour of continuous operation	1. (a) Fully charge the battery (b) Run the vehicle until the battery dies (c) Record the duration

Continued on next page

Table 8 – continued from previous page

Item	Requirement	Test/calibration procedure
Power consumption	1. Peak current is lower than 6A 2. Average current is lower than 2A	1. (a) Connect a 0.25Ω 10W sensing resister in the battery loop (b) Use AVR32's differential ADC channel to probe the voltage drop on the resister (c) Run the vehicle for 10 minutes and send the ADC readings real-timely to the host computer (d) Find the peak current among the data 2. In the above test, continue to calculate the average of all data

3.2 Integrated Testing

The functionality of the whole vehicle is tested by the following procedure:

1. Prepare several ramps with different slopes
2. Test the stability when run one wheel over the ramp
3. Use the same ramp and run several times with two, three, or four wheels over it
4. Try turning the vehicle on the ramp
5. Change the slope of ramp and repeat step 2 to step 4
6. Add different load on the vehicle and repeat step 2 to step 5
7. Prepare an uneven surface with random small pits and hills
8. Run the vehicle on the surface to see if it can adapt to drastic change

3.3 Tolerance Analysis

1. If the slope of the surface is in a great magnitude, the vehicle will fail to adapt the slope. The reason is that when the vehicle raise its wheel, the latter one will reach its maximum hight before the former one cross the slope; thus, no more room exists for further adaption. Furthermore, the chassis may also have contact with the slope. Since surfaces with slope higher than 50° is rare, we decide to set the maximum slope at 50° during test.

   We find the extreme slope by the following procedure:

(a) Run the car at 15cm/s

(b) Gradually increase the slope of the ramp until the car fails to run over it

2. Velocity also places significant effect on the performance. When operating on a high velocity, the vehicle achieves stability more difficultly since the active suspension system requires time to adjust itself. In our design and testing, the maximum velocity used is around 15cm/s.

   We find the extreme speed by the following procedure:

   (a) Fix the slope of ramp at 20°

   (b) Gradually increase the speed of the car

   (c) Import the potentiometer readings to MATLAB

   (d) Read the peak of the position data, if the peak is under 18 degree, record the speed as the extreme

4 Ethical Considerations

We commit ourselves to the IEEE code of ethics, the ethical concerns we have is listed as following:

1. 'to accept responsibility in making decisions consistent with the safety, health and welfare of the public, and to disclose promptly factors that might endanger the public or the environment;'
   In the case of our project, our design is intended for laboratory use. Nonetheless, we cannot neglect the possibility that our design is used by unauthorized person or group for transporting unexpected and dangerous chemical, for instance, any sort of poisons or explosives. If our product is to be used, we will make sure we only hand it to authorized person or group, and make sure only they know how to operate the system.

2. 'to be honest and realistic in stating claims or estimates based on available data;'
   We will test and simulate our project strictly and make sure all the data we provide are real. In addition we will not exaggerate our system's performance. All statements about performance and functionality will only be made after the design is thoroughly tested.

3. 'to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others;'
   We will accept any constructional suggestion from anyone, including engineers, technicians, and people without technical background. We seek to improve our design to maximize its utility and minimize any potential threat to society. We will also credit anyone who has offered us help in the realization of this project.

Besides from the IEEE code of ethics, we have our own standards in working as well. We will make sure we think deeply about the impacts of our decision before we make one. We will strive to make this a satisfying project.

5 Cost and Schedule

5.1 Cost

1. **Labor:**

   Zhangxiaowen Gong: \$35/hr × 12hr/week × 10 weeks = \$4200

   Wenjia Zhou: \$35/hr × 12hr/week × 10 weeks = \$4200

   Jun Ma: \$35/hr × 12hr/week × 10 weeks = \$4200

   Total: \$4200 × 3 = \$12600

2. **Parts:**

Table 9: Itemized budget

Part name	**Unit cost**	**comment**	**Number required**	**Subtotal**
AT32UC3C2256	\$13.78	http://search.digikey.com/us/en/products/AT32UC3C2256C-A2UR/AT32UC3C2256C-A2URCT-ND/2774159	1	13.78
ATmega8A	\$1.84	from China	4	\$7.36
ATmega32U4	\$3.38	from China	1	\$3.38
Worm motor	storage		4	N/A
Servo	storage		4	N/A
DC motor	storage		4	N/A
Encoder	\$2		4	\$8
Battery	storage		1	N/A
nRF24L01+ RF module	storage		2	N/A
MPU-6050	\$15.00	http://invensense.com/mems/gyro/mpu6050.html	1	\$15.00
HMC5883L	\$3.79	http://search.digikey.com/us/en/products/HMC5883L-TR/342-1082-1-ND/2507853	1	\$3.79
IRF3205S	\$0.42	from China	16	\$6.72
IR2104	\$0.62	from China	8	\$4.96
MAX629	\$1.54	from China	1	\$1.54
LM2596S-5.0	\$0.31	from China	4	\$1.24
LM1117-3.3	\$0.05	from China	2	\$0.10

Continued on next page

Table 9 – continued from previous page

Part name	**Unit cost**	**comment**	**Number required**	**Subtotal**
LM2594-3.3	$1.54	from China	1	$1.54
OLED Display	$14.95	http://www.sparkfun.com/products/10133	1	$14.95
oscillator	$0.77	from China	6	$4.62
PCB	$20		3	$60
Misc. components	N/A	N/A	N/A	$10
Total				$156.98

The total expense is $12600 + $156.98 = $12756.98

Note: machine shop cost is not covered.

5.2 Schedule

Table 10: Schedule

Time	**Task**	**People in duty**
2.5-2.11	Write proposal	Jun, Wenjia, Andy
2.12-2.18	Build the chassis and mechanical part of the car	Wenjia
	Order components	Andy
2.19-2.25	Derive the transfer function of motor	Jun
2.26-3.3	PCB design	Andy
	Implement Kalman filter	Wenjia
	Implement Complimentary filter	Jun
3.4-3.10	Simulate the design using MATLAB	Wenjia
	Solder and test PCBs	Andy
3.11-3.17	Tune PID parameter to reach the desired performance	Jun
	Compare the output between Kalman filter and Complimentary filter	Wenjia
	Implement device drivers	Andy
3.18-3.24	Implement the PID control in C	Wenjia
	Implement the wireless communication and RTOS based embedded programs	Andy
3.25-3.31	Verify the design on the car	Andy
4.1-4.7	Test the design under different scenario	Jun
4.8-4.14	Tune PID parameter again if needed	Wenjia

References

[1] Albert-Jan Baerveldt and Robert Klang.
Low-cost and low-weight attitude estimation system for an autonomous helicopter.
In *IEEE International Conference on Intelligent Engineering Systems, Proceedings, INES*, page 391∼395, 1997.
Cited By (since 1996): 43.

[2] Powell J. David Franklin, Hene F. and Abbas Emami-Naeini.
Feedback Control of Dynamic Systems, page 43.
Pearson Education, Inc., 5 edition, 2006.

[3] Mohinder S. Grewal and Angus P. Andrews.
Linear Dynamic Systems, page 31∼66.
John Wiley & Sons, Inc., 2008.

[4] InvenSense Inc.
Mpu-6000 and mpu-6050 product specification revision 3.2, 2011.

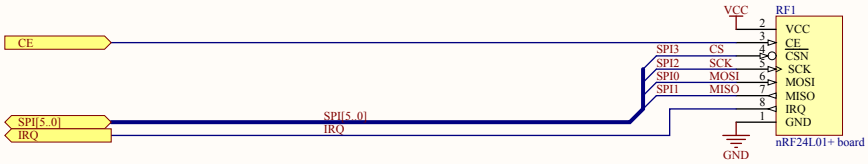
[5] Maxim Integrated Products.
Max629 datasheet rev 1.

[6] Philips Semiconductors.
Bi-directional level shifter for i^2c-bus and other systems, 1997.

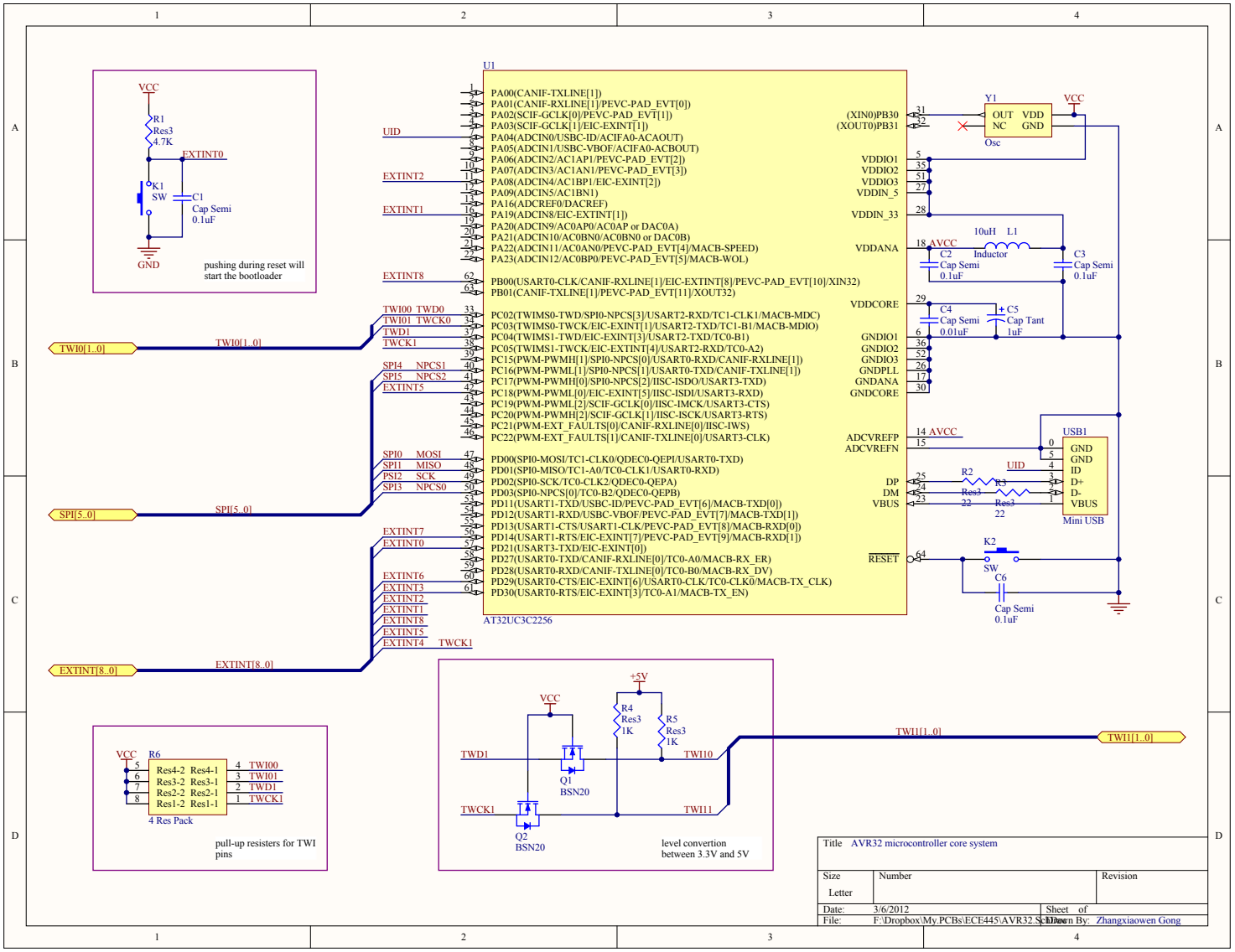
List of Figures

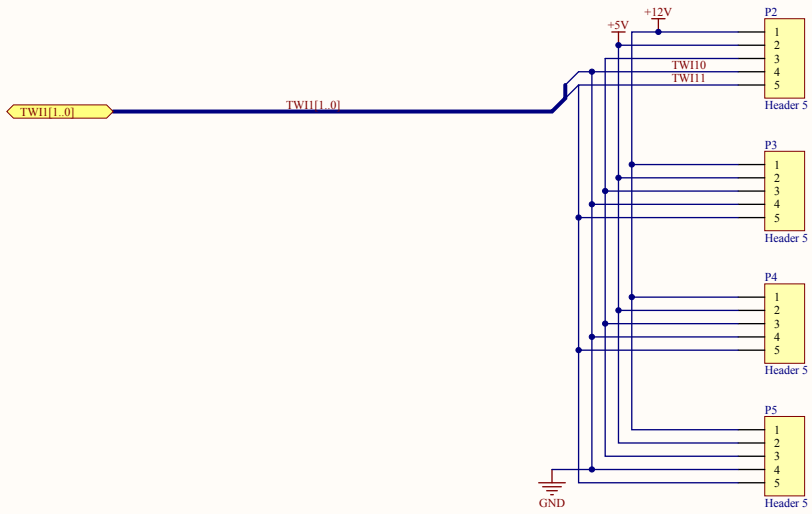
List of Tables

A Schematics for the main controller

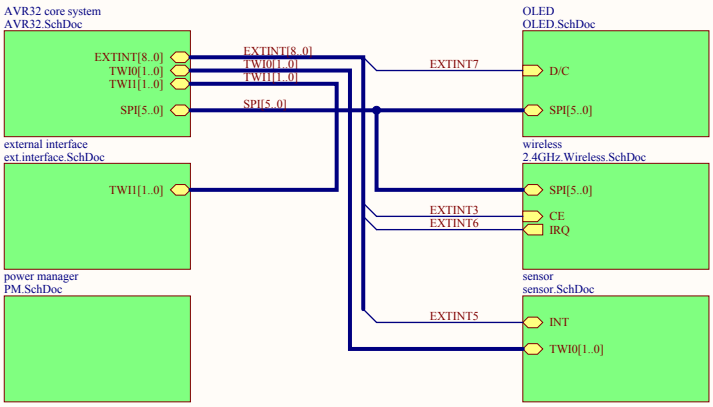


Title	2.4GHz wireless interface		
Size: Letter	Number		Revision
Date:	3/6/2012	Sheet	of
File:	F:\Dropbox\..\2.4GHz.Wireless.SchDoc	Drawn By:	Zhangxiaowen Gong

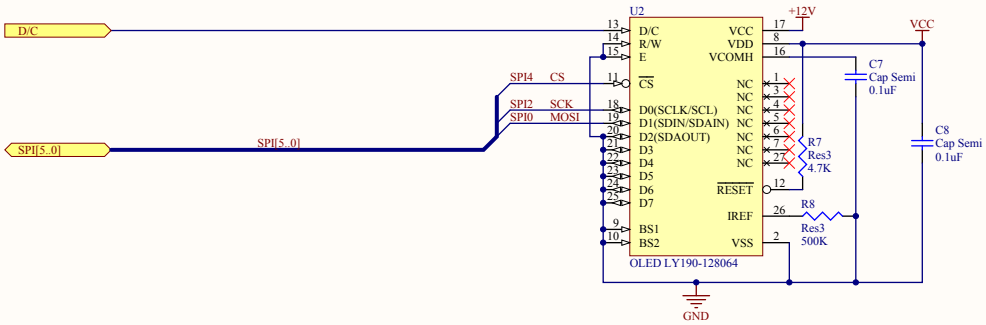

U1
AT32UC3C2256
PA00(CANIF-TXLINE[1])
PA01(CANIF-RXLINE[1]/PEVC-PAD_EVT[0])
PA02(SCIF-GCLK[0]/PEVC-PAD_EVT[1])
PA03(SCIF-GCLK[1]/EIC-EXINT[1])
PA04(ADCIN0/USBC-ID/ACIFA0-ACAOUT)
PA05(ADCIN1/USBC-VBOF/ACIFA0-ACBOUT)
PA06(ADCIN2/AC1AP1/PEVC-PAD_EVT[2])
PA07(ADCIN3/AC1AN1/PEVC-PAD_EVT[3])
PA08(ADCIN4/AC1BP1/EIC-EXINT[2])
PA09(ADCIN5/AC1BN1)
PA16(ADCREF0/DACREF)
PA19(ADCIN8/EIC-EXTINT[1])
PA20(ADCIN9/AC0AP0/AC0AP or DAC0A)
PA21(ADCIN10/AC0BN0/AC0BN0 or DAC0B)
PA22(ADCIN11/AC0AN0/PEVC-PAD_EVT[4]/MACB-SPEED)
PA23(ADCIN12/AC0BP0/PEVC-PAD_EVT[5]/MACB-WOL)
PB00(USART0-CLK/CANIF-RXLINE[1]/EIC-EXTINT[8]/PEVC-PAD_EVT[10]/XIN32)
PB01(CANIF-TXLINE[1]/PEVC-PAD_EVT[11]/XOUT32)
PC02(TWIMS0-TWD/SPI0-NPCS[3]/USART2-RXD/TC1-CLK1/MACB-MDC)
PC03(TWIMS0-TWCK/EIC-EXINT[1]/USART2-TXD/TC1-B1/MACB-MDIO)
PC04(TWIMS1-TWD/EIC-EXINT[3]/USART2-TXD/TC0-B1)
PC05(TWIMS1-TWCK/EIC-EXTINT[4]/USART2-RXD/TC0-A2)
PC15(PWM-PWMH[1]/SPI0-NPCS[0]/USART0-RXD/CANIF-RXLINE[1])
PC16(PWM-PWML[1]/SPI0-NPCS[1]/USART0-TXD/CANIF-TXLINE[1])
PC17(PWM-PWMH[0]/SPI0-NPCS[2]/IISC-ISDO/USART3-TXD)
PC18(PWM-PWML[0]/EIC-EXINT[5]/IISC-ISDI/USART3-RXD)
PC19(PWM-PWML[2]/SCIF-GCLK[0]/IISC-IMCK/USART3-CTS)
PC20(PWM-PWMH[2]/SCIF-GCLK[1]/IISC-ISCK/USART3-RTS)
PC21(PWM-EXT_FAULTS[0]/CANIF-RXLINE[0]/IISC-IWS)
PC22(PWM-EXT_FAULTS[1]/CANIF-TXLINE[0]/USART3-CLK)
PD00(SPI0-MOSI/TC1-CLK0/QDEC0-QEPI/USART0-TXD)
PD01(SPI0-MISO/TC1-A0/TC0-CLK1/USART0-RXD)
PD02(SPI0-SCK/TC0-CLK2/QDEC0-QEPA)
PD03(SPI0-NPCS[0]/TC0-B2/QDEC0-QEPB)
PD11(USART1-TXD/USBC-ID/PEVC-PAD_EVT[6]/MACB-TXD[0])
PD12(USART1-RXD/USBC-VBOF/PEVC-PAD_EVT[7]/MACB-TXD[1])
PD13(USART1-CTS/USART1-CLK/PEVC-PAD_EVT[8]/MACB-RXD[0])
PD14(USART1-RTS/EIC-EXINT[7]/PEVC-PAD_EVT[9]/MACB-RXD[1])
PD21(USART3-TXD/EIC-EXINT[0])
PD27(USART0-TXD/CANIF-RXLINE[0]/TC0-A0/MACB-RX_ER)
PD28(USART0-RXD/CANIF-TXLINE[0]/TC0-B0/MACB-RX_DV)
PD29(USART0-CTS/EIC-EXINT[6]/USART0-CLK/TC0-CLK0/MACB-TX_CLK)
PD30(USART0-RTS/EIC-EXINT[3]/TC0-A1/MACB-TX_EN)
(XIN0)PB30
(XOUT0)PB31
VDDIO1
VDDIO2
VDDIO3
VDDIN_5
VDDIN_33
VDDANA
VDDCORE
GNDIO1
GNDIO2
GNDIO3
GNDPLL
GNDANA
GNDCORE
ADCVREFP
ADCVREFN
DP
DM
VBUS
RESET
UID
EXTINT2
EXTINT1
EXTINT8
TWI00 TWD0
TWI01 TWCK0
TWD1
TWCK1
SPI4 NPCS1
SPI5 NPCS2
EXTINT5
SPI0 MOSI
SPI1 MISO
PSI2 SCK
SPI3 NPCS0
EXTINT7
EXTINT0
EXTINT6
EXTINT3
EXTINT2
EXTINT1
EXTINT8
EXTINT5
EXTINT4 TWCK1
TWI0[1..0]
SPI[5..0]
EXTINT[8..0]
Y1
OUT VDD
NC GND
Osc
VCC
10uH L1
Inductor
C2 Cap Semi 0.1uF
C3 Cap Semi 0.1uF
C4 Cap Semi 0.01uF
C5 Cap Tant 1uF
AVCC
USB1
GND
GND
ID
D+
D-
VBUS
Mini USB
R2
R3
Res3
22
K2
SW
C6 Cap Semi 0.1uF
R1 Res3 4.7K
EXTINT0
K1
SW
C1 Cap Semi 0.1uF
GND
pushing during reset will start the bootloader
R6
Res4-2 Res4-1
Res3-2 Res3-1
Res2-2 Res2-1
Res1-2 Res1-1
4 Res Pack
TWI00
TWI01
TWD1
TWCK1
pull-up resisters for TWI pins
+5V
R4 Res3 1K
R5 Res3 1K
TWD1
TWI10
Q1 BSN20
TWCK1
TWI11
Q2 BSN20
level convertion between 3.3V and 5V
TWI1[1..0]
Title AVR32 microcontroller core system
Size Letter
Number
Revision
Date: 3/6/2012
Sheet of
File: F:\Dropbox\My.PCBs\ECE445\AVR32.SchDoc
Drawn By: Zhangxiaowen Gong



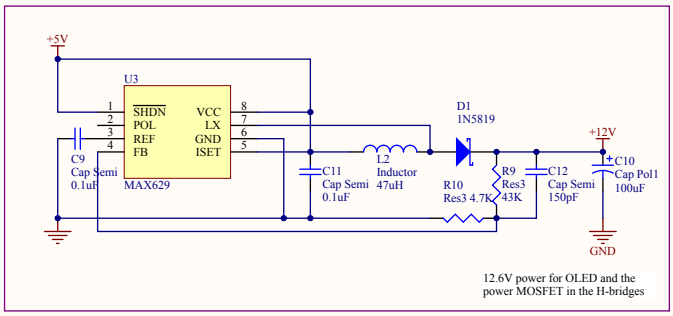
Title	External interface to motor control boards		
Size: Letter	Number		Revision
Date:	3/6/2012	Sheet	of
File:	F:\Dropbox\..\ext.interface.SchDoc	Drawn By:	Zhangxiaowen Gong

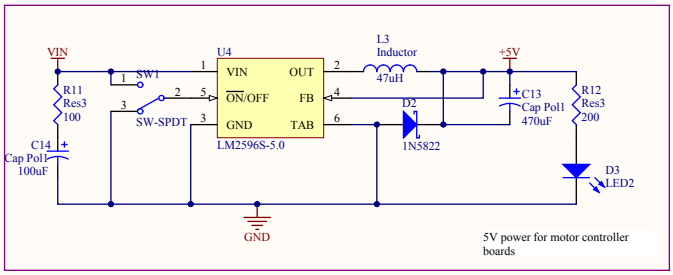


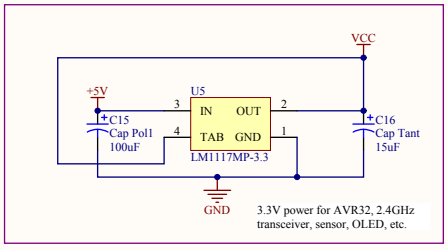
Title	Main sheet		
Size: Letter	Number		Revision
Date:	3/6/2012	Sheet	of
File:	F:\Dropbox\..\main.board.SchDoc	Drawn By:	Zhangxiaowen Gong



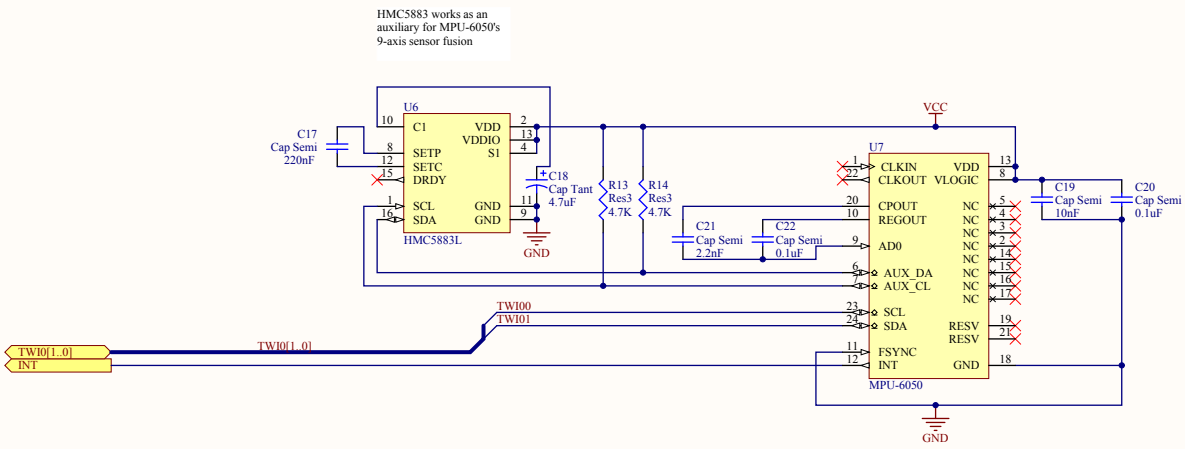
Title	0.9" OLED display interface	
Size: Letter	Number	Revision
Date: 3/6/2012	Sheet of	
File: F:\Dropbox\My.PCBs\ECE445\OLED.SchDoc	Drawn By: Zhangxiaowen Gong	







Title: Power management providing +3.3V, +5V, and +12V		
Size: Letter	Number:	Revision:
Date: 3/6/2012	Sheet of	
File: F:\Dropbox\My.PCBs\ECE445\PM.SchDoc	Drawn By: Zhangxiaowen Gong	



Title	6-axis gryoscope and accelerometer interface	
Size: Letter	Number	Revision
Date: 3/6/2012	Sheet of	
File: F:\Dropbox\My.PCBs\ECE445\sensor.Sch	Drawn By: Zhangxiaowen Gong	

Bill of Materials

<Parameter Title not found>

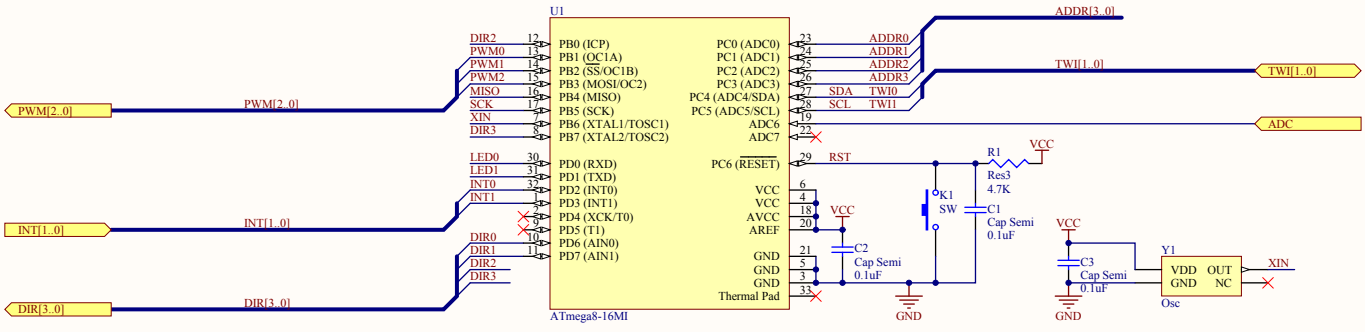
Source Data From: ECE445.PrjPCB
Project: ECE445.PrjPCB
Variant: None

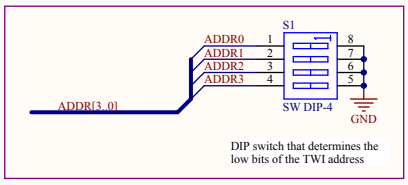
Creation Date: 3/6/2012 5:32:23 AM
Print Date: 06-Mar-12 5:32:26 AM

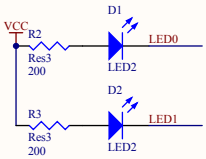
Footprint	Comment	LibRef	Designator	Description	Quantity
C0805	Cap Semi	Cap Semi	C1, C2, C3, C4, C6, C7, C8, C9, C11, C12, C17, C19, C20, C21, C22	Capacitor (Semiconductor SIM Model)	15
TC3216-1206	Cap Tant	Cap Pol1	C5, C16, C18	Polarized Capacitor (Radial)	3
CAPT-6.3*8	Cap Pol1	Cap Pol1	C10, C13, C14, C15	Polarized Capacitor (Radial)	4
SMB	1N5819	D Schottky	D1	Schottky Diode	1
SMB	1N5822	D Schottky	D2	Schottky Diode	1
CD3216-1206	LED2	LED2	D3	Typical RED, GREEN, YELLOW, AMBER GaAs LED	1
Button1	SW	SW-PB	K1, K2	Switch	2
INDC4510-1804	Inductor	Inductor	L1	Inductor	1
SMD2010	Inductor	Inductor	L2, L3	Inductor	2
2*1 3.96mm Connector	Header 2	Header 2	P1	Header, 2-Pin	1
PH2.0 - 5	Header 5	Header 5	P2, P3, P4, P5	Header, 5-Pin	4
SOT23-3L	BSN20	BSN20	Q1, Q2	N-channel Enhancement Mode FET	2
6-0805_N	Res3	Res3	R1, R2, R3, R4, R5, R9, R10, R11, R12	Resistor	9
EXB38V	4 Res Pack	4 Res Array	R6		1
J1-0603	Res3	Res3	R7, R8, R13, R14	Resistor	4
nRF24L01	nRF24L01+ board	nRF24L01	RF1		1
MSS-22C02	SW-SPDT	SW-SPDT	SW1	SPDT Subminiature Toggle Switch, Right Angle Mounting, Vertical Actuation	1
VQFN64_M	AT32UC3C2256	AT32UC3C2256	U1		1
	OLED LY190-128064	OLED12864	U2		1
SO8_L	MAX629	MAX629	U3		1
TS5B_N	LM2596S-5.0	LM2596S-5.0	U4	SIMPLE SWITCHER Power Converter 150 KHz 3A Step-Down Voltage Regulator	1
MP04A_N	LM1117MP-3.3	LM1117MP-3.3	U5	800mA Low-Dropout Linear Regulator	1
LPCC16	HMC5883L	HMC5883L	U6	Honeywell 3-Axis Digital Compass IC	1
	MPU-6050	MPU_6050	U7		1
mini_USB	Mini USB	Mini_USB	USB1		1
Osc	Osc	Osc	Y1	Osc	1
					62

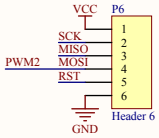
Approved	Notes	

B Schematics for the side controllers

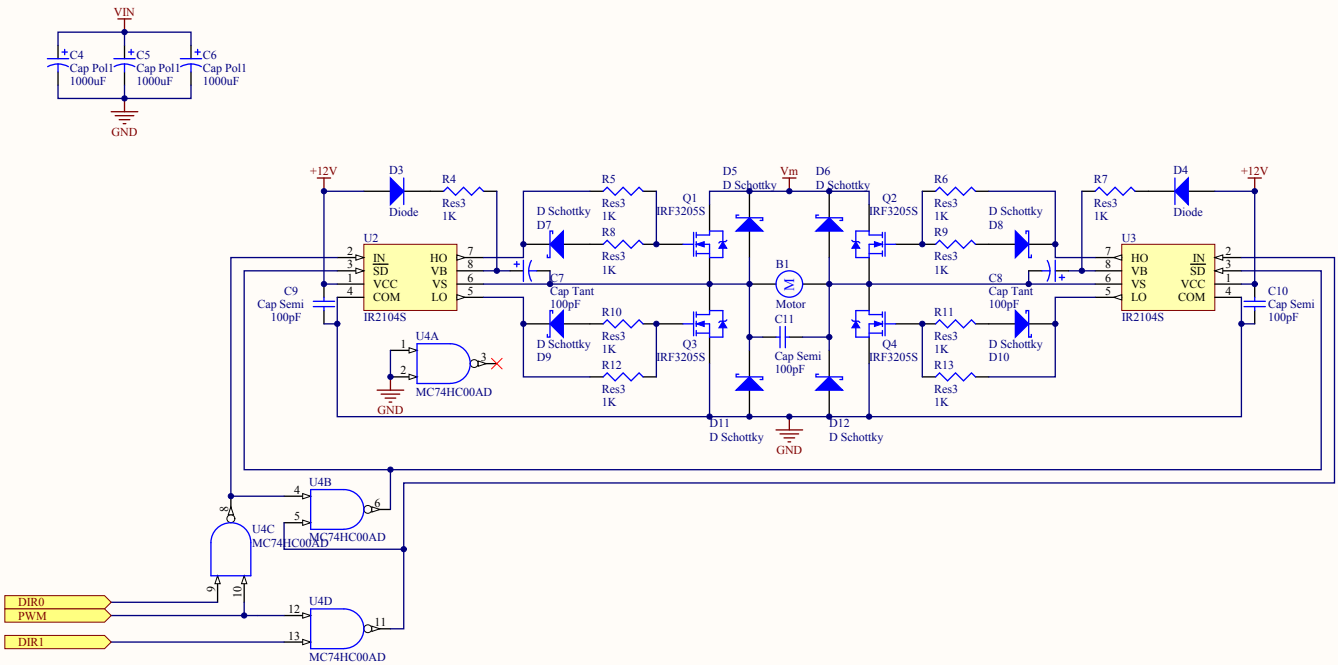




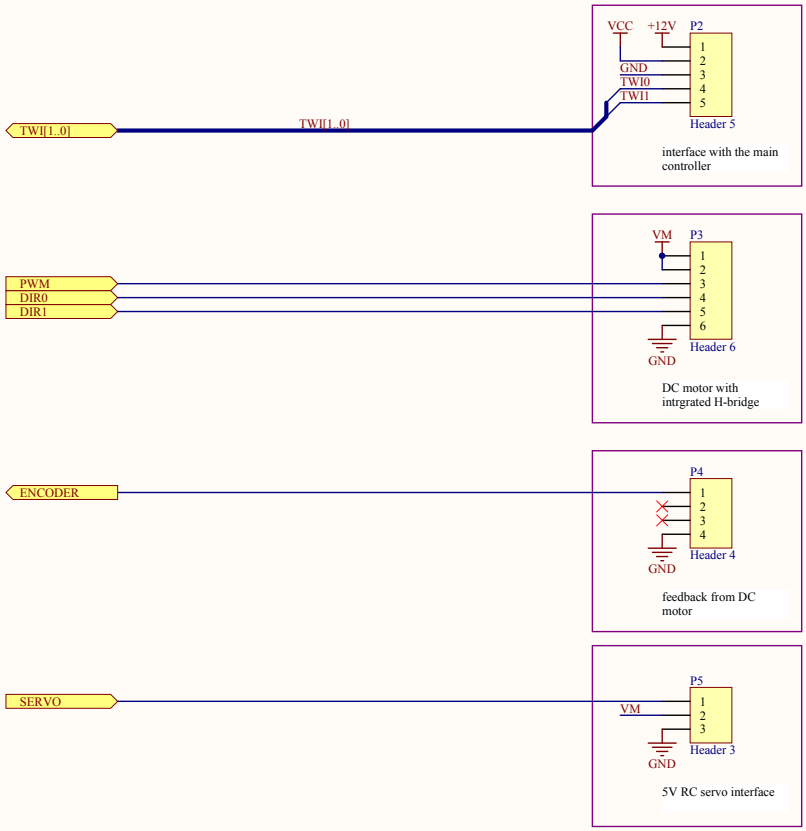




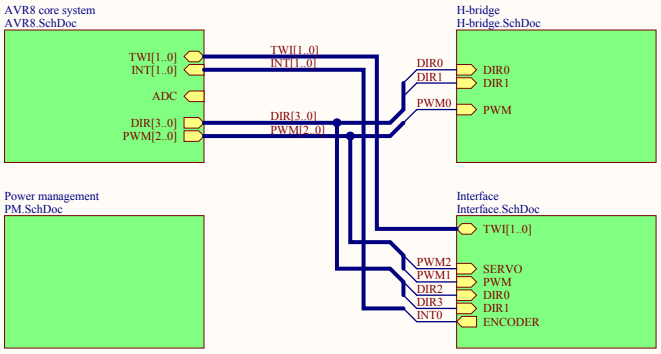
Title AVR8 microcontroller core system		
Size Letter	Number	Revision
Date: 3/6/2012		Sheet of
File: F:\Dropbox\..\AVR8.SchDoc		Drawn By: Zhangxiaowen Gong



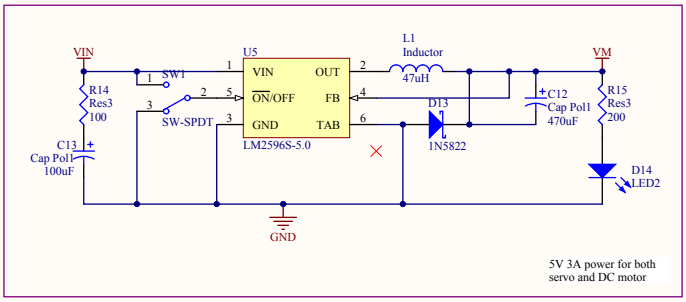
Title	H-bridge DC motor driver		
Size: Letter	Number		Revision
Date:	3/6/2012	Sheet	of
File:	F:\Dropbox\..\H-bridge.SchDoc	Drawn By:	Zhangxiaowen Gong



Title	Interface with motor, servo, and main controller	
Size: A4	Number	Revision
Date: 3/6/2012		Sheet of
File: F:\Dropbox\..\Interface.SchDoc		Drawn By: Zhangxiaowen Gong



Title	Main sheet		
Size: Letter	Number		Revision
Date:	3/6/2012	Sheet	of
File:	F:\Dropbox\..\motor.controller.SchDoc	Drawn By:	Zhangxiaowen Gong



Title: Power management providing +5V		
Size: Letter	Number:	Revision:
Date: 3/6/2012		Sheet of
File: F:\Dropbox\..\PM.SchDoc		Drawn By: Zhangxiaowen Gong

Bill of Materials

<Parameter Title not found>

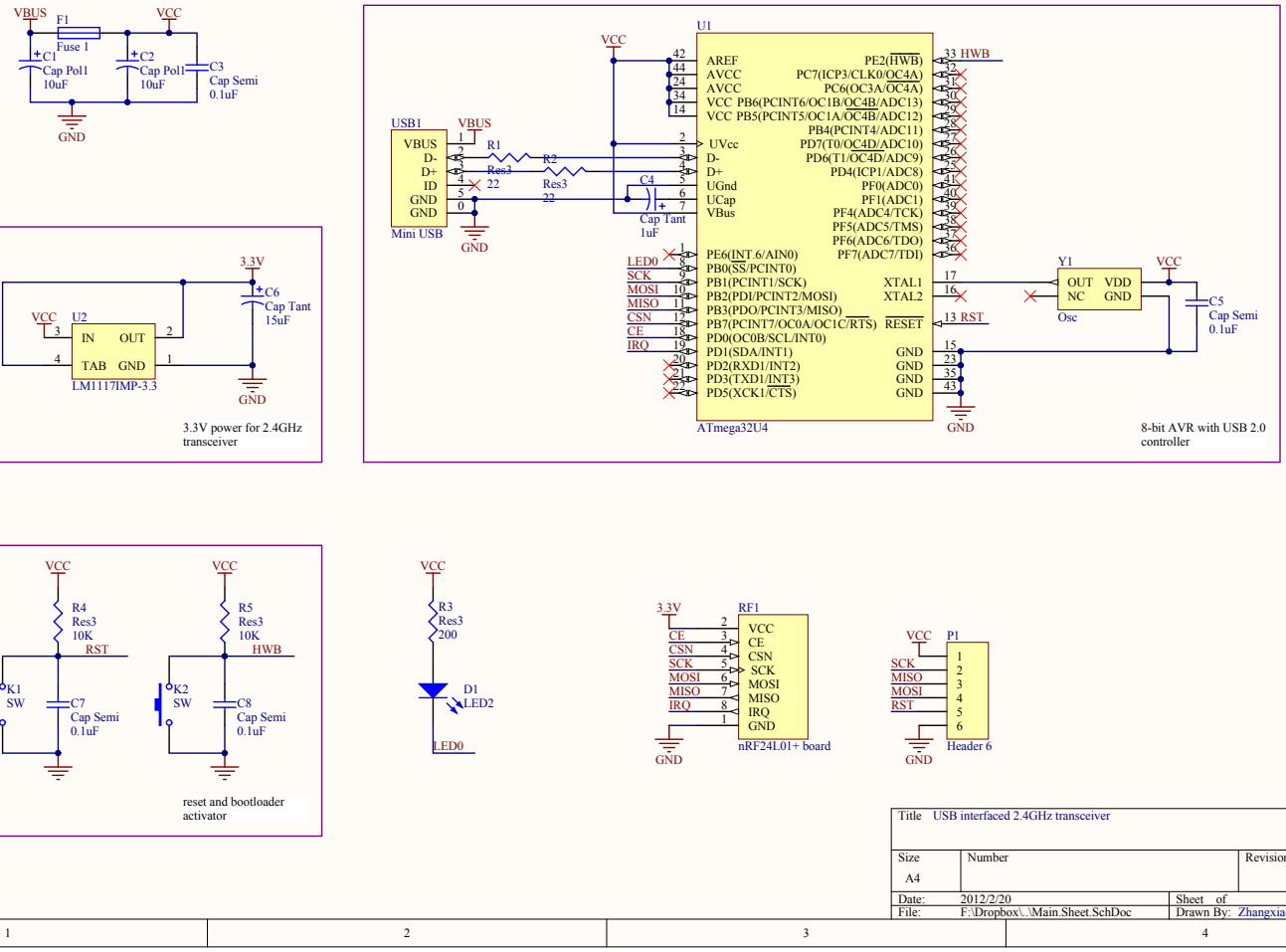
Source Data From: ECE445.motor.controller.PrjPcb
Project: ECE445.motor.controller.PrjPcb
Variant: None

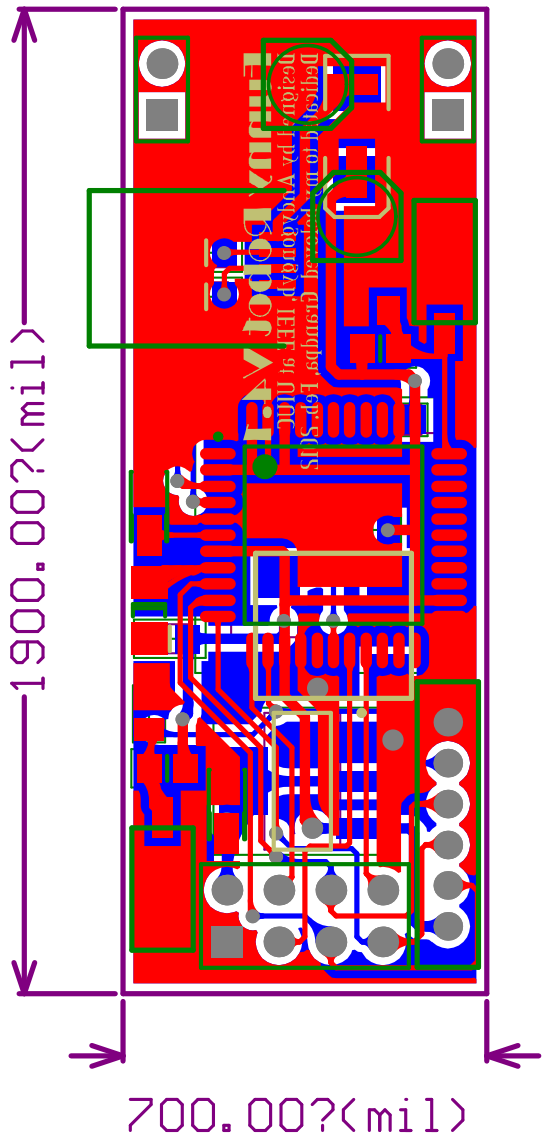
Creation Date: 3/6/2012 5:29:57 AM
Print Date: 06-Mar-12 5:30:04 AM

Footprint	Comment	LibRef	Designator	Description	Quantity
2*1 3.96mm Connector	Motor	Motor	B1	Motor, General Kind	1
C0805	Cap Semi	Cap Semi	C1, C9, C10	Capacitor (Semiconductor SIM Model)	3
CC2012-0805	Cap Semi	Cap Semi	C2, C3	Capacitor (Semiconductor SIM Model)	2
CAPT-10*10	Cap Pol1	Cap Pol1	C4, C5, C6	Polarized Capacitor (Radial)	3
TC3216-1206	Cap Tant	Cap Pol1	C7, C8	Polarized Capacitor (Radial)	2
C1206	Cap Semi	Cap Semi	C11	Capacitor (Semiconductor SIM Model)	1
CAPT-6.3*8	Cap Pol1	Cap Pol1	C12, C13	Polarized Capacitor (Radial)	2
CD3216-1206	LED2	LED2	D1, D2, D14	Typical RED, GREEN, YELLOW, AMBER GaAs LED	3
SMB	Diode	Diode	D3, D4	Default Diode	2
SMC	D Schottky	D Schottky	D5, D6, D11, D12	Schottky Diode	4
SMB	D Schottky	D Schottky	D7, D8, D9, D10	Schottky Diode	4
SMB	1N5822	D Schottky	D13	Schottky Diode	1
Button1	SW	SW-PB	K1	Switch	1
SMD2010	Inductor	Inductor	L1	Inductor	1
2*1 3.96mm Connector	Header 2	Header 2	P1	Header, 2-Pin	1
PH2.0 - 5	Header 5	Header 5	P2	Header, 5-Pin	1
HDR1X6	Header 6	Header 6	P3	Header, 6-Pin	1
HDR1X4	Header 4	Header 4	P4	Header, 4-Pin	1
HDR1X3	Header 3	Header 3	P5	Header, 3-Pin	1
PH2.0 - 6	Header 6	Header 6	P6	Header, 6-Pin	1
D2PAK	IRF3205S	IRF3205S	Q1, Q2, Q3, Q4	HEXFET Power MOSFET	4
6-0805_N	Res3	Res3	R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15	Resistor	15
SO8_N	SW DIP-4	SW DIP-4	S1	DIP Switch, 4 Position, SPST	1
MSS-22C02	SW-SPDT	SW-SPDT	SW1	SPDT Subminiature Toggle Switch, Right Angle Mounting, Vertical Actuation	1
32M1-A_N	ATmega8-16MI	ATmega8-16MI	U1	8-Bit AVR Microcontroller with 8K Bytes of In-System Programmable Flash Memory	1
SO-8_N	IR2104S	IR2104S	U2, U3	High and Low Side Driver	2
751A-02_N	MC74HC00AD	MC74HC00AD	U4	Quad 2-Input NAND Gate	1
TS5B_N	LM2596S-5.0	LM2596S-5.0	U5	SIMPLE SWITCHER Power Converter 150 KHz 3A Step-Down Voltage Regulator	1
Osc	Osc	Osc	Y1	Osc	1
					63

Approved	Notes	

C Schematics and PCB for the PC resceiver


VBUS
F1
Fuse 1
VCC
C1
Cap Pol1
10uF
C2
Cap Pol1
10uF
C3
Cap Semi
0.1uF
GND
3.3V
C6
Cap Tant
15uF
VCC
U2
IN
OUT
TAB
GND
LM1117IMP-3.3
GND
3.3V power for 2.4GHz transceiver
U1
VCC
AREF
AVCC
AVCC
VCC
VCC
UVcc
D-
D+
UGnd
UCap
VBus
PE6(INT.6/AIN0)
PB0(SS/PCINT0)
PB1(PCINT1/SCK)
PB2(PDI/PCINT2/MOSI)
PB3(PDO/PCINT3/MISO)
PB7(PCINT7/OC0A/OC1C/RTS)
PD0(OC0B/SCL/INT0)
PD1(SDA/INT1)
PD2(RXD1/INT2)
PD3(TXD1/INT3)
PD5(XCK1/CTS)
PE2(HWB)
PC7(ICP3/CLK0/OC4A)
PC6(OC3A/OC4A)
PB6(PCINT6/OC1B/OC4B/ADC13)
PB5(PCINT5/OC1A/OC4B/ADC12)
PB4(PCINT4/ADC11)
PD7(T0/OC4D/ADC10)
PD6(T1/OC4D/ADC9)
PD4(ICP1/ADC8)
PF0(ADC0)
PF1(ADC1)
PF4(ADC4/TCK)
PF5(ADC5/TMS)
PF6(ADC6/TDO)
PF7(ADC7/TDI)
XTAL1
XTAL2
RESET
GND
GND
GND
GND
HWB
RST
ATmega32U4
USB1
VBUS
D-
D+
ID
GND
GND
Mini USB
R1
Res3
22
R2
Res3
22
C4
Cap Tant
1uF
LED0
SCK
MOSI
MISO
CSN
CE
IRQ
Y1
OUT
NC
VDD
GND
Osc
C5
Cap Semi
0.1uF
8-bit AVR with USB 2.0 controller
VCC
R4
Res3
10K
RST
K1
SW
C7
Cap Semi
0.1uF
R5
Res3
10K
HWB
K2
SW
C8
Cap Semi
0.1uF
reset and bootloader activator
VCC
R3
Res3
200
D1
LED2
LED0
3.3V
RF1
CE
CSN
SCK
MOSI
MISO
IRQ
VCC
CE
CSN
SCK
MOSI
MISO
IRQ
GND
nRF24L01+ board
GND
VCC
P1
SCK
MISO
MOSI
RST
1
2
3
4
5
6
Header 6
GND
Title USB interfaced 2.4GHz transceiver
Size A4
Number
Revision
Date: 2012/2/20
Sheet of
File: F:\Dropbox\..\Main.Sheet.SchDoc
Drawn By: Zhangxiaowen Gong


1900.00?(mil)
700.00?(mil)

Bill of Materials

<Parameter Title not found>

Source Data From: **USB.To.Wireless.PrjPcb**

Project: **USB.To.Wireless.PrjPcb**

Variant: **None**

Creation Date: 2012/2/20 12:33:52

Print Date: 20-Feb-12 12:33:54 PM

Footprint	Comment	LibRef	Designator	Description	Quantity
CAPT-4*5	Cap Pol1	Cap Pol1	C1, C2	Polarized Capacitor (Radial)	2
C0805	Cap Semi	Cap Semi	C3, C5, C7, C8	Capacitor (Semiconductor SIM Model)	4
TC3216-1206	Cap Tant	Cap Pol1	C4, C6	Polarized Capacitor (Radial)	2
CD3216-1206	LED2	LED2	D1	Typical RED, GREEN, YELLOW, AMBER GaAs LED	1
SMB	Fuse 1	Fuse 1	F1	Fuse	1
Button1	SW	SW-PB	K1, K2	Switch	2
PH2.0 - 6	Header 6	Header 6	P1	Header, 6-Pin	1
6-0805_N	Res3	Res3	R1, R2, R3, R4, R5	Resistor	5
nRF24L01	nRF24L01+ board	nRF24L01	RF1		1
TQFP44_N	ATmega32U4	ATmega32U4	U1		1
MP04A_N	LM1117IMP-3.3	LM1117IMP-3.3	U2	800mA Low-Dropout Linear Regulator	1
mini_USB	Mini USB	Mini_USB	USB1		1
Osc	Osc	Osc	Y1	Osc	1
					23

Approved	Notes	