Club is dedicated to giving students hands-on experience and access to parts and equipment free of charge. We meet on Tuesdays for tutorials on theory (electronics, signals, embedded programming, design tools, etc.) with application to a semester long project. On Thursdays we meet to work with sensors and of-the-shelf parts to assemble robots and other cool stuff that we can go to competitions with. You can join or drop in any time. No commitment or notification is required. See our <a href="Schedule.html" target="\_self">Schedule</a> for more details.

Practice Sessions:

On Thursdays you will be able to take any available off-the-shelf parts like Xbee modules, IR sensors, Bluetooth transmitters, etc. and build anything with them. Particularly, you can build robots and participate in competitions in spring, after exams. Registration fees, flights, hotels other expenses in Calgary or New York are paid completely by HEC.

List of competitions we prepare and go in Winter2015 Semester:

Freescale Cup (<https://community.freescale.com/docs/DOC-1284>)

Western Canadian Robot Games (<http://www.robotgames.com/>)

As well, during both semesters we are collaborating with Architecture Department and help Architecture students create interactive projects that go to Nuit Blanche, Toronto Offsite Festival, Gladstone Hotel exhibition “Come Up to My Room” and more. See previous collaborative projects here and here.

List of projects that are proposed by competitions or by Architecture department. (Link to the previous collaboration. See examples

<http://ryearchdesignlab.blogspot.ca/2014/01/more-imagery-of-some-of-325-church.html>

<http://ryearchdesignlab.blogspot.ca/search/label/GrowOp2014> )

Tutorials: This semester we will work our way to gain knowledge and skills to control a dc motor with microcontroller from very scratch. By the end of the semester one would have their own designed motor driver shield that will work with both Arduino and ARM Cortex M0+ MCU based board. This will involve learning theory about motor types and their control, transistors, H-bridges, voltage regulation, charge pumps, PCB CAD layout, prototyping techniques, Arduino and introduction to bare-metal programming on ARM, which will be continued in Winter semester. On the week of November 17th we are joined by the engineer from the semiconductor industry. He will talk about current industry demands and directions as well about microcontroller architecture.

Tutorials are free of charge and run on Tuesdays at 6:00pm in ENG 303. See our <a href="Schedule.html" target="\_self">Schedule</a>. You will easily catch up regardless of when you decide to join. It is highly recommended to bring your laptops. Tutorial notes if any will be uploaded to the <a href="Tutorials\_and\_Sessions.html">Tutorials and Sessions</a> the day after our weekly meeting. Upcoming tutorial links are empty.

Contact us:

Club runs on Tuesdays and Thursdays 6:00pm in ENG 303, 245 Church Street.

Email us to [hecryerson@gmail.com](mailto:hecryerson@gmail.com)

Anna Leshchenko, Elecrical Engineering [anna.leshchenko@ryerson.ca](mailto:anna.leshchenko@ryerson.ca)

Sami Dalati, Electrical Engineering [sami.aldalati@ryerson.ca](mailto:sami.aldalati@ryerson.ca)

We are collaborating with Ryerson Architecture Department, namely with <a href=" http://www.arch.ryerson.ca/vincent-hui/ " target="\_self"> Vincent Hui </a> and his <a href=" http://ryearchdesignlab.blogspot.ca/search/label/%5BR%5DED%5BU%5DX%20LAB" target="\_self"> [R]ed[U]x Lab </a>.

Our supervisor is <a href=" http://www.ee.ryerson.ca/~jasmith/ " target="\_self"> Dr. James Smith </a>.

Tutorial 1 Fast tracking through Arduino.

The objective of the Fall2014 semester long project and schedule will be outlined. The objective is to design and build the motor driver board from scratch and learn to control motors with it. This tutorial is designed to give you the idea what the final result should be. You will be given Arduino board and off the shelf Motor Shield for it and asked to put all components together to make motor spin. Example codes will be provided. You will be explained basic principles of hardware operation, software side of Ardiono IDE. Concepts of interrupts will be explained as well and now you will be asked to write a simple Arduino program with interrupts and try it on the board. Download Arduino IDE: <http://arduino.cc/en/Main/Software>

Tutorial 2 Sept 30th

Finishing writing program with Interrupts if you didn’t do so on the previous tutorial. Concepts of motor operation. Types of motors. (<https://community.freescale.com/docs/DOC-1284>) methods of dc motor control. . If there is a force opposing the motor, then the terminals are short circuited and the current through the terminals can go as high as 14 A. The Motor has a Resistance between 0.9 and 1.0 ohm. or more.<https://community.freescale.com/docs/DOC-1067> H bridges half bridges: on chip, on board. What are supplementary ccts and components necessary: protection ccts, charge pumps and so on. Again. Use Arduino and h bridges on the chip, assemble the cct, and write the program to spin the motor. You can put motors on chasse and make it run.

Servos: https://community.freescale.com/docs/DOC-1027

Tutorial 3 Oct 7th

We will be looking into general schematics of an H-bridge based on BJTs and MOSFETs. Before proceeding with building it we will look at transistor operation principles, solve couple of problems and assemble the circuit to drive LED with MOSFET operating as a switch.

H-bridges on transistors. An H-Bridge circuit has a control circuit, usually PWM, which then determines the switching of high-voltage supply to drive a current. Typical embedded H-Bridges can drive about 5A of current.  You can place H-Bridges in parallel to balance the current load. For example, if you place two 5A (peak) H-Bridge outputs in parallel, the system can support up to 10A current. General chematics. But first, lets explain bjts and mosfets. Ccts with them . couple of problems. Assemble useful cct. And try to provide enough current to the led. Keep it Cool. H-Bridge's dissipate A LOT of heat. Heat = increases inefficiency of a semiconductor, so the better job you do keeping it cool, the better (and longer) it will work for you.

Tutorial 4 Oct 14th

We will be looking at several problems involving all of those components. You will assemble and test the example circuit to drive LED with PWM output of Arduino.

http://www.completepowerelectronics.com/comparison-of-mosfet-with-bjt/

Power mos, op amp and charge pump. Useful ccts. t has very low ON resistance and no junction voltage drop when forward biased. These features make MOSFET an extremely attractive power supply switching device. Power Electronics Design Engineers prefer-MOSFET over BJT in their applications. MOSFETs are majority carrier device, means flow of current inside the device is carried out either flow of electrons(N-Channel MOSFET) or flow of holes(P-Channel MOSFET). So when the device turns off, reverse recombination process won't happen. It leads to short turn ON/OFF times. As switching time is less, loss associated with it less.  
So for high frequency applications, where the switching loss is major impact in total power loss of the circuit, this device is the right choice.  
On the other hand, for applications having lower operating frequencies BJT is superior. MOSFET is voltage controlled device whereas BJT is current controlled device.  
Mosfet operation is controlled by gate-source voltage(VGS). In BJT, the operation is controlled by base current. As we know that providing constant voltage is easier than providing constant current in electrical circuits. So Mosfet's gate drive circuit is less complex than BJT's base drive circuit. the main current flow is controlled by an electrostatic field. An FET has the great advantage that no current flows into the control input (called the gate), the main current is turned on and off by the level of voltage on the gate.

Tutorial 5 Oct 21st

Assemble H-bridge with protection, and charge pump.

Tutorial 6 Oct 28th

Before coming to the session download Eagle Cad on your computers: <http://www.cadsoftusa.com/download-eagle/?language=en> . You will be making schematic of your board together in Eagle Cad and placing components on the PCB.

Tutorial 7 Nov 4th Prototyping Techniques

The modern PCB design will be discussed: why electromagnetism matters. You will solder or wire wrap your layout.

Tutorial 8 Nov 11th Trying the board with Arduino.

You will run your motor drive with Ardiono program. You can attach it all to the car chassis and build your RC car with one of our Xbee modules.

Tutorial 9 Nov 18th Intro to ARM and KDS

Introduction to ARM architecture and IDE for it. We will discuss the components needed for programming any chip of any architecture. We will look into the Winter2015 semester and schedule. We are also having Derrick Klotz from Freescale Semiconductors to talk about your engineering future. You don’t have to download it if you don’t want but the Kinetis Design Studio (IDE for ARM Freescale MCU) will be used next semester (<http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=KDS_IDE> )

Tutorial 10th Nov 25th General Academics

I will be answer any questions up to my knowledge and abilities on current academic curriculum: Signals I, DSP, Circuits, Power Electronics, Digital Logic, Electromagnetics and so on.

Tutorial 1

The objective of the semester long project and schedule will be outlined.

--why this project was chosen . Industry demands—motors and controls. In automotive: fan, engine, timing, windows. Cd drivers, hard discs..

Fast tracking through Arduino.

--board explanation, ide run down: wiring language, libraries, concepts of mcu programming.

--Motorshields from different manufacturers. Library or example download.

--whats on the board and why: The voltage or current that must be delivered to the motor to work is too much for a microcontroller output port so an intermediary device must be used.

--download library or use example. When done, explain interrupts and ask to write the program that runs motor one direction for 2 sec, then blinks RED led once, then runs motor different direction for 3 sec and after blinks Green LED-repeat the cycle.

**Interrupts.**

On-chip peripheral systems generate maskable interrupts, which are recognizedonly if the global interrupt mask bit (I) in the condition code register (CCR) is clear. Maskable interrupts are prioritized according to a default arrangement. A few important interrupt sources that are always enabled are called non-maskable interrupts.When interrupt conditions occur in an on-chip peripheral system, an interrupt status flag is set to indicate the condition. When the user’s program has properly responded to this interrupt request, the status flag must be cleared.

**Reset** is used to force the microcontroller unit (MCU) to assume a set of initial conditions and to begin executing instructions from a predetermined starting address. After reset, the CPU fetches the restart vector from locations $FFFE, FFFF. During the first three cycles and begins executing instructions. The stack pointer and other CPU registers are indeterminate immediately after reset.

The CPU in a microcontroller sequentially executes instructions. In many applications, it is necessary to execute sets of instructions in response to requests from various peripheral devices. These requests are often asynchronous to the execution of the main program. Interrupts provide a way to temporarily suspend normal program execution so the CPU can be freed to service these requests. After an interrupt has been serviced,

the main program resumes as if there had been no interruption. The instructions executed in response to an interrupt are called the interrupt service routine. These routines are much like subroutines except that they are called through the automatic hardware interrupt mechanism rather than by a subroutine call instruction, and all CPU

registers are saved on the stack rather than just saving the program counter. The interrupt logic then pushes the contents of all CPU registers onto the stack so the CPU context can be restored after the interrupt is finished. After stacking the CPU registers, the vector for the highest priority pending interrupt source is loaded into the program

counter, and execution continues with the first instruction of the interrupt service routine. An interrupt is concluded with a return-from-interrupt (RTI) instruction, which causes all CPU registers and the return address to be recovered from the stack so that the interrupted program can resume as if there had been no interruption. An interrupt can be recognized at any time provided it is enabled by its local mask (if any) and by the global mask bit in the CCR. Once any interrupt source is recognized, the CPU will respond at the completion of the currently executing instruction. Instructions cannot be interrupted; rather, the CPU decides whether to fetch another instruction or process an interrupt. In calculating the latency time from the actual interrupt request to the CPU response to that request, the user must consider the possibility that the CPU had just started a long instruction as the interrupt was requested. Most instructions are two to four cycles long, but the multiply (MUL) and integer divide (IDIV) or fractional divide (FDIV) instructions are 10 and 41 cycles, respectively. When an interrupt has been serviced as needed, the return-from interrupt (RTI) instruction terminates interrupt processing and returns to the program that was running at the time of the interruption. During servicing of the interrupt, some or all of the CPU registers will have changed. To continue the former program as if it had not been interrupted, the registers must be restored to the values present at the time the former program was interrupted. The RTI instruction accomplishes this by pulling (loading) the saved register values from the stack memory. The last value to be pulled from the stack is the program counter, which causes processing to resume where it was interrupted. Most interrupt sources in the M68HC11 have separate interrupt vectors; thus, there is usually no need for software to poll control registers to determine the cause of an interrupt. The maskable interrupt sources respond to a fixed-priority relationship except that any one source can be dynamically elevated to the highest priority position of any maskable source.The I bit in the CCR acts as a primary enable control for all maskable interrupts. When the I bit is set, interrupts can become pending but will not be honored. When the I bit is clear, interrupts are enabled to interrupt normal program flow when an interrupt source requests service. The I bit is set during reset to prevent interrupts from being honored until minimum system initialization has been performed. Part of this minimum initialization would be to load the stack pointer so it points to an appropriate area of RAM.

Every enty in a vector table is a 32 bit pointer.

In risk- index addressing vs cisk absolute addressing. Opcodes don’t support absolute addressing.

Tutorial 2 Sept 30th

Finishing writing program with Interrupts if you didn’t do so on the previous tutorial. Concepts of motor operation.

Types of motors: DC, Servo, Stepper. See page DC and Servos.

Type of drivers: 3 phase, relay and H-bridges. Control: open loop, closed loop. We need to measure DC Bus voltage, Back-EM voltage, phase currents, DC Bus current, heatsink temperature, We need to measure speed and rotor position from different sensors (hall sensors, quadrature encoder, tacho generator, sin/cos interface, etc.)

If there is a force opposing the motor, then the terminals are short circuited and the current through the terminals can go as high as 14 A. The Motor has a Resistance between 0.9 and 1.0 ohm. or more.

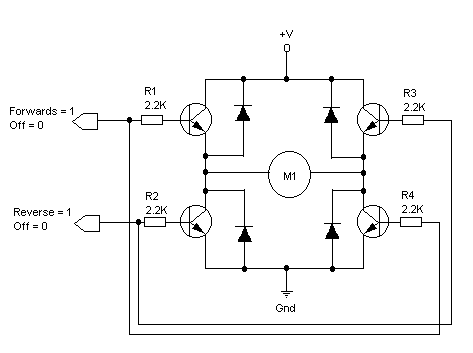
H bridges half bridges: on chip, on board. Show the circuit and explain the IC. Assemble and run with Arduino.

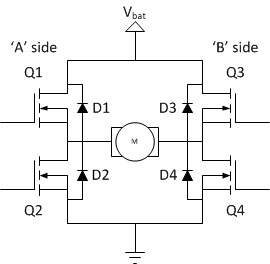
You can attach it all to the car chassis and make it run.

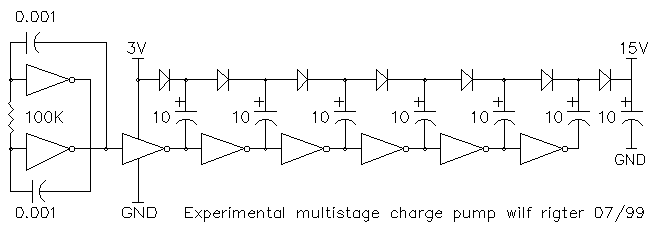
Tutorial 3 Oct 7th

H-bridges on transistors. An H-Bridge circuit has a control circuit, usually PWM, which then determines the switching of high-voltage supply to drive a current. Typical embedded H-Bridges can drive about 5A of current. For example, if you place two 5A (peak) H-Bridge outputs in parallel, the system can support up to 10A current. Keep it Cool. H-Bridge's dissipate A LOT of heat. Heat = increases inefficiency of a semiconductor, so the better job you do keeping it cool, the better (and longer) it will work for you.

General chematics:



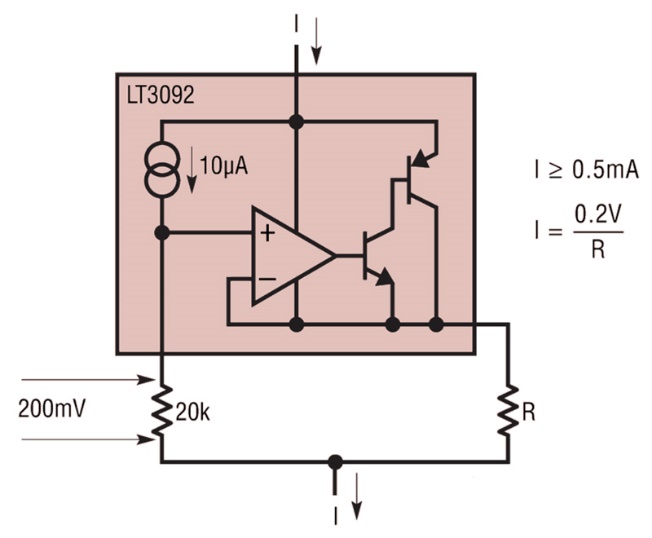


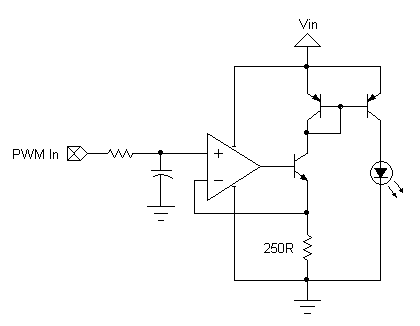


Problems: bjt, mosfet and their general operation: 2 blank pages (A Y notes) and power mosfet led driver with Arduino. Assemble run.

Tutorial 4 Oct 14th

Couple of problems:





My cct: blank page:

MOSFET is voltage controlled device whereas BJT is current controlled device. Mosfet operation is controlled by gate-source voltage(VGS). In BJT, the operation is controlled by base current. As we know that providing constant voltage is easier than providing constant current in electrical circuits. So Mosfet's gate drive circuit is less complex than BJT's base drive circuit. the main current flow is controlled by an electrostatic field. An FET has the great advantage that no current flows into the control input (called the gate), the main current is turned on and off by the level of voltage on the gate.