Electromyogram Controlled RC Car

Jessica Matthews
Grant Hunter
German Gomez Urbina
Emil Jafarli

Summary of Project:
Control a RC car via muscle signals. Muscle activities are translated into data by two EMGs that are then decoded into meaningful commands for the RC car.

Abstract

The controller we created used two electromyogram (EMG) sensors to gather the user's muscle electric signals. The system analyzed the user's data and sent the appropriate signals to a remote control car. We successfully opened up the transmitter for the remote control car. We found that to make the car respond via software there are four contacts that need to have the ability to connect to the negative terminal of the 9V battery. We used NPN transistors as switches. Which were then connected to the DE0 Nano and successfully sent the car commands via our software. First we built one EMG and found we could get three levels of freedom that could control the forward, reverse and stop motions. We then built a second EMG to control the right and left turn motions. We attached one EMG to each arm and successfully controlled the RC car using muscle contractions in the forearm.

Declaration of Original Content

The design elements of this project and report are entirely the original work of the authors and have not submitted for credit in any other course except as follows:

-DE0 nano FPGA board provided by Altera[1]
-Schematic for EMG sensor provided by Brian Kaminski through Instructables[2]
-49Mhz transmitter and RC car provided by Tyco R/C[3]
-Nios II CPU core provided by Altera[4]
-SDRAM core provided by Altera[5]
-SRAM core provided by Altera[6]
-Clock core provided by Altera[7]
-BJT as an electronic switch[8]

Table of contents

Functional Requirements of Project

This project utilizes signals from two Electromyograms to control an RC Car. Electrodes are placed on the users arm in three places. One placed at the end of the muscle on the forearm, one placed in the middle of the muscle on the forearm and one placed just below the elbow on the bone for a point of reference. Gestures from the user translate into very small voltage differences between the two electrodes placed on the forearm. These voltage differences are measured, amplified, rectified, sent through low pass filters and are converted from analog signals to digital signals. These signals are then decoded in software into a command sent via a RF transmitter to RC car. Based on the movements of the user, an RC car is powered and controlled. From each of the EMGs there are three voltage levels. Zero voltage, which is achieved when the muscle is resting, a middle range voltage which occurs when the muscle is partially engaged and a maximum voltage which is the voltage we receive from the EMGs when the muscle is fully engaged. For the first EMG these three voltage levels control the stop, forward and reverse commands. Stop occurs when the muscle is resting, reverse when the muscle is partially engaged and forward occurs when the muscle is fully engaged. For the second EMG, no turn signal occurs when the muscle is resting, the left turn occurs when the muscle is partially engaged and then right turn occurs when the muscle is fully engaged. Software determines the level of the muscle's engagement and translates that to a command which turn on a specified transistor switch to send the appropriate command to the RC Car. This is done through a 4 bit bus, each bit represents one control on the transmitter, forward, backwards, left and right. In order to prevent unwanted commands such as forward and back at the same time or left and right at the same time, software reads the previous signal on the bus and clears the two bits to be updated before assigning the new command.

These functional requirements were met and for a short period of time we were able to successfully control the RC Car using the EMGs. You can see a short video of the project working at https://www.youtube.com/watch?v=F1XCiN0wbIU. We found that the INA 106 ICs in our EMGs could be blown up very easily and we determined this is what caused our EMGs to send unwanted signals to the DE0 Nano.

Design and description of operation

Overall operation

EMG controlled RC car is a project of collecting the voltages from the muscles, converting and classifying them to specific control commands, and based on those commands controlling the RC car. To get started on this project, first we need to create a system with custom components. Figure 1 shows the hardware block diagram of our system and Hardware Design section gives more in-depths analysis of each component and its usage.The EMG signals are sent through the ADC channels of DE0 nano board. In our software design, we have 2 tasks controlling and classifying the received signals. Both tasks have the same exact layout with the exception of one being forward/reverse control and the other one right/left control. The flowchart for the algorithm could be seen from Figure 2.The core of the algorithm could be explained as following.

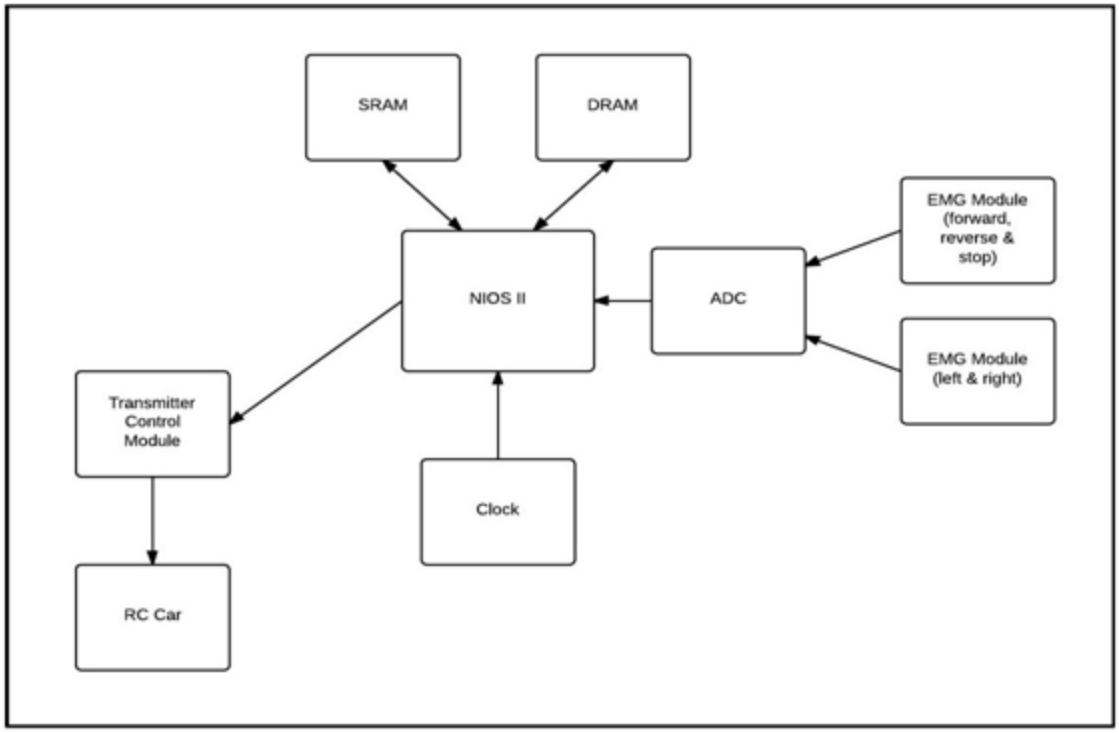


Figure 1 : Hardware Diagram Showing Data Flow Between Components

Adaptive Thresholding Algorithm

Due to the volatility in voltage variation the algorithm takes two samples in a very short period of time. The samples are then compared to see if they are increasing. If so, the bigger sample is compared to the forward threshold value. If greater we'll go to the forward mode directly. Otherwise another sample is collected, and then it’s compared with the previous one to see if it is increasing, and if it is, we will do the aforementioned steps again.

If the collected samples are not increasing, then we are not on the rising edge of signals, which means that we can right away compare the predefined threshold values with the samples and execute the associated command. Once a specific mode is determined, no further processing is required. More samples are collected to ensure that the signal does not fall below the stop threshold value.

After we have collected the EMG signals and determined the command the user wants, we have to send that signal to the RC car. As explained above, we use four bit bus to set a particular bit, which in turn completes the circuit between transistor and transmitter and causes the car to move in a specific direction.

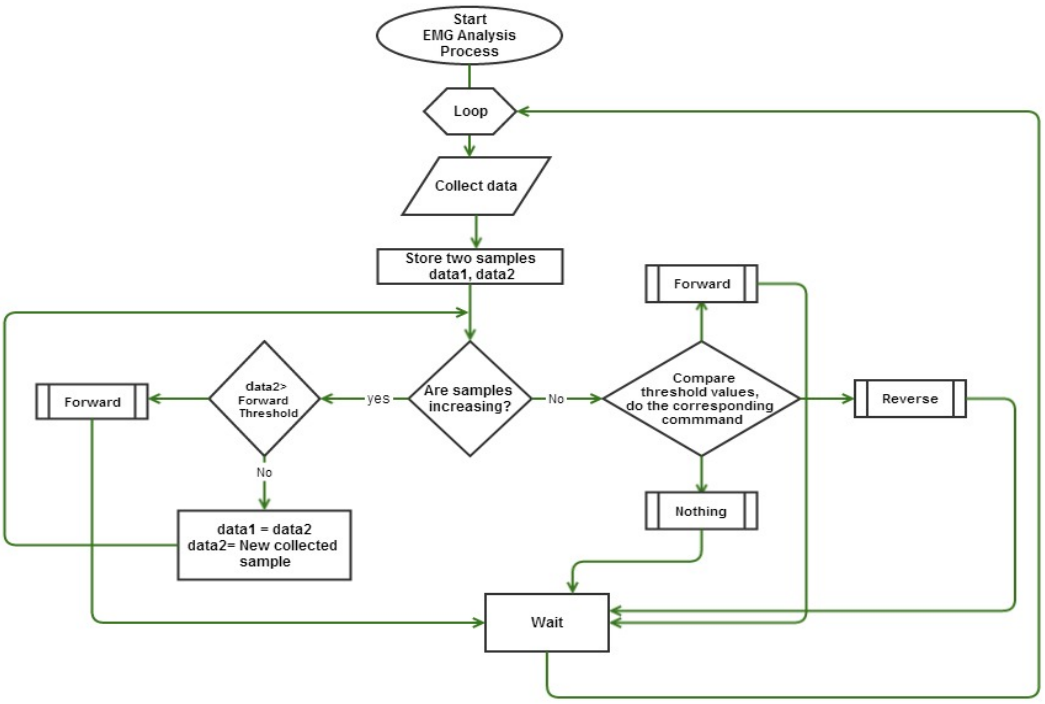


Figure 2: EMG Signal Classification Algorithm

Hardware Design

Altera DE0 Nano FPGA Board

The Altera DE0 Nano board is used as the microcontroller unit of this project. The following components of it were used in our design:

• Altera Cyclone IV EP4CE22F17C6 FPGA
• ADC – used for reading and converting emg signals
• LEDs - to display any kind of data we want
• 32MB SDRAM – used for flashing the code
• 2x13 header with ADC pins -used for connecting the emg circuit
• GPIO_0 header – for transferring the signals to the transmitter

The following components were used to create our Qsys system in Quartus II 12.1.
• NiosII/e CPU
• System ID Peripheral
• JTAG UART
• Interval Timer
• SDRAM
• Clock Bridge
• PIO (Parallel I/O)
• EPCS Serial Flash Controller
• ADC
• LEDS

Bill of materials

Component: MCU

- Part: Altera DE0 Nano

Quantity: 1 Cost per unit: $132.11 Total: $132.11 Supplier: University
Datasheet:
http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Components/General%20IC/PS-MPU-6000A.pdf
Website: https://www.sparkfun.com/products/11028

Component: EMG Sensor

- Part: TL072CP IC Part No: ADS1210U-ND

Quantity: 6 Cost per unit: $0.728 Total: $4.368 Supplier : Digikey
Datasheet: http://www.ti.com/lit/ds/symlink/tl071a.pdf
Website: http://www.digikey.com/product-detail/en/TL072CP/296-1775-5-ND/277421
Order Status: Out for delivery

- Part: INA106 IC Part No: ADS1210U-ND

Quantity: 2 Cost per unit: $11.39 Total: $22.78 Supplier : Digikey
Datasheet: http://www.ti.com/lit/ds/symlink/tl071a.pdf
Website: http://www.digikey.com/product-detail/en/TL072CP/296-1775-5-ND/277421
Order Status: Out for delivery

- Part: EMG Cables

Quantity: 6 Cost per unit: $30.08 (3pkg) Total: $60.16 Supplier : bio-medical
Website:
http://bio-medical.com/products/din-ekg-emg-eeg-snap-leads-40-inch-3-lead-kit-iec-standard.html

- Part: EMG Electrodes

Quantity: 6 Cost per unit: $30.08 Total: $30.08 (3pkg) Supplier : bio-medical
Website:
http://bio-medical.com/products/covidien-kendall-disposable-surface-emg-ecg-ekg-electrodes-1-24mm-50pkg.html

- Part: 9V Battery

Quantity: 2 Cost per unit: $4.99 Total: $9.98 (3pkg) Supplier : Best Buy
Website: http://www.bestbuy.ca/en-CA/category/9-volt-batteries/22673.aspx

- Part: 1.0uF Tant Capacitor Part No: 399-3529-ND

Quantity: 4 Cost per unit: $0.71 Total: $2.84 Supplier : Digikey
Datasheet:http://www.kemet.com/kemet/web/homepage/kechome.nsf/vapubfiles/F3100_TaDipRad.pdf/$file/F3100_TaDipRad.pdf
Website:
http://www.digikey.com/product-search/en?WT.z_header=search_go&lang=en&site=us&keywords=399-3529-ND&x=0&y=0

- Part: 0.01uF Ceramic Disc Part No: 490-3812-ND

Quantity: 2 Cost per unit: $0.19 Total: $0.38 Supplier : Digikey
Datasheet: http://www.murata.com/products/catalog/pdf/c49e.pdf
Website:
http://www.digikey.com/product-search/en?WT.z_header=search_go&lang=en&site=us&keywords=490-3812-ND&x=0&y=0

- Part: 1.0uF Ceramic Disc Part No: 399-4389-ND

Quantity: 2 Cost per unit: $1.12 Total: $2.24 Supplier : Digikey
Datasheet:
http://www.kemet.com/kemet/web/homepage/kechome.nsf/vapubfiles/F3101_GoldMax.pdf/$file/F3101_GoldMax.pdf
Website:
http://www.digikey.com/product-search/en?WT.z_header=search_go&lang=en&site=us&keywords=399-4389-ND&x=0&y=0

- Part: 150kOhm 1% Part No: 150KXTR-ND

Quantity: 6 Cost per unit: $0.10 Total: $0.60 Supplier : Digikey
Datasheet: http://www.yageo.com/documents/recent/General%20Type_MFR.pdf
Website: http://www.digikey.com/product-detail/en/MFR-25FBF52-150K/150KXBK-ND/13508

- ·Part: 1MOhm 1% Part No: RNF14FTD1M00TR-ND

Quantity: 4 Cost per unit: $0.15 Total: $0.60 Supplier : Digikey
Datasheet: http://www.seielect.com/Catalog/SEI-RNF_RNMF.pdf
Website:
http://www.digikey.com/product-detail/en/RNF14FTD1M00/RNF14FTD1M00CT-ND/1975201

- Part: 80.6kOhm 1% Part No: RNF14FTD80K6TR-ND

Quantity: 4 Cost per unit: $0.15 Total: $0.60 Supplier : Digikey
Datasheet: http://www.seielect.com/Catalog/SEI-RNF_RNMF.pdf
Website:
http://www.digikey.com/product-detail/en/RNF14FTD80K6/RNF14FTD80K6CT-ND/1975154

- Part: 10kOhm 1% Part No: 10.0KXTR-ND

Quantity: 12 Cost per unit: $0.10 Total: $1.20 Supplier : Digikey
Datasheet: http://www.seielect.com/Catalog/SEI-RNF_RNMF.pdf
Website: http://www.digikey.com/product-detail/en/MFR-25FBF52-10K/10.0KXBK-ND/13219

- Part: 100 kOhm Trimmer Part No: D4AA15-ND

Quantity: 2 Cost per unit: $0.54 Total: $1.08 Supplier : Digikey
Datasheet: http://industrial.panasonic.com/www-data/pdf/AOH0000/AOH0000CE2.pdf
Website:
http://www.digikey.com/product-search/en?WT.z_header=search_go&lang=en&site=us&keywords=D4AA15-ND&x=0&y=0

- Part: 1 kOhm 1% Part No: RNF14FTD1K00TR-ND

Quantity: 2 Cost per unit: $0.15 Total: $0.30 Supplier : Digikey
Datasheet: http://www.seielect.com/Catalog/SEI-RNF_RNMF.pdf
Website:
http://www.digikey.com/product-detail/en/RNF14FTD1K00/RNF14FTD1K00CT-ND/1975018

- Part: 1N4148 Diode Part No: 1N4148FS-ND

Quantity: 4 Cost per unit: $0.1 Total: $0.4 Supplier : DigikeyDatasheet:
http://www.fairchildsemi.com/ds/1N/1N914.pdf
Website:
http://www.digikey.com/product-search/en?WT.z_header=search_go&lang=en&site=us&keywords=1N4148FS-ND&x=0&y=0

Component: Transmitter

- Part: 49Mhz transmitter

Quantity: 1 Cost per unit: $(included in RC car) Supplier : University

- Part: transistors

Quantity: 4 Cost per unit: <$1 Total: <$4 Supplier : University

Component: Misc

- Part: header pins/connectors

Quantity: NA Cost per unit: <$5 Supplier : University

- Part: RC Car

Quantity: 1 Cost per unit: $50 Total: $50 Supplier : University

- Part: 40-pin ribbon cable

Quantity: 1 Cost per unit: $10 Total: $10 Supplier : University

- Part: perf board

Quantity: 1 Cost per unit: $7.50 Total: $7.50 Supplier : University

The total cost is $341.218. (The items supplied by university also included which has a total cost of $203.61)

Different Available Sources of Reusable Design Units

Adaptive thresholding
For any future project the adaptive thresholding algorithm C code is available in github:
Source code:https://github.com/grantdhunter/ECE

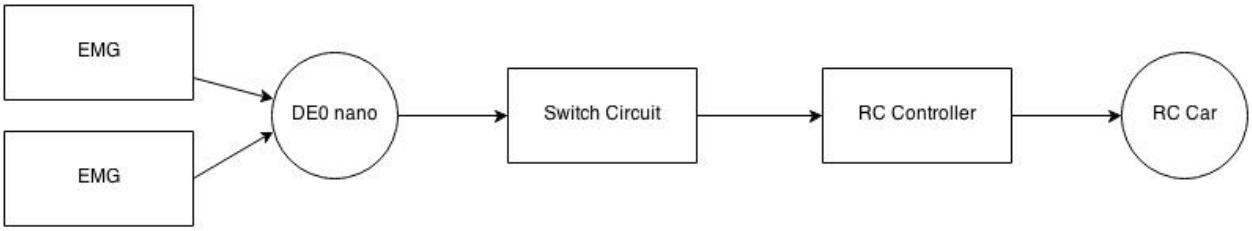
DE0 Nano ADC
We used the Verilog and C file provided by the manufacturer for the ADC on the DE0 board. Still, the provided code had a bug on it since it failed to properly collect data from multiple channels. We fixed the problem by keeping the first 12-bit of the data and masking all other bits to zero. We also made an appnote about it and all the source codes, system files can be dowloaded from the following link:
https://www.ualberta.ca/~delliott/local/ece492/appnotes/2014w/G6_ADC/

DE0 Nano SDRAM
We also used the QSYS and Verilog file provided by the manufacturer for the SDRAM. It properly explained how to fix the clock-skew problem which enabled the proper access of the SDRAM chip.

How to flash to DE0-Nano tutorial:
Provided by the lab TA Jessica, this tutorial was very easy to follow and got the job done.

Datasheet

User perspective of the EMG-Controlled RC Car:



The IO signals describing our FPGA top design is as follows:

Signal	Description
clk_clk	System clock, at 100MHz
reset_reset_n	Reset clock that is connected to a PLL.
sdram_*	Signals that enable the SDRAM
epcs_*	Signals for non-volatile subsystem
pio_led_green_export	Connects signal to the LEDs
transmitter_export	Signal that connects system to pin GPIO_0
adc_*	Signals to enable the ADC subsystem

Below are the voltages and currents from each subsystem:

Subsystem	Current (A)	Voltage (V)
EMGs	0.1	3.1
DE0-nano	--	3.3V
RC Transmitter (49Hz)	--	9V
RC Car	--	9V

Clock of the FPGA-core: 100MHz.

It can be assumed that the EMGs' power consumption is constant at about 3.1[V]*0.1[A] = 0.31[W]. The RC car, transmitter, and switch circuit power consumption is negligible in comparison, whilst the DE0 power consumption is constant at about 1.88[W].
That value is calculated below:

Altera Core Power: 3.3[V]*0.019[A] + 1.2[V] * 0.654[A] = 0.8475[W]
EPCS16 Power: 3.3[V] * 0.02[A] = 0.066[W]
SDRAM Power: 3.3[V] * 0.292[A] = 0.9636[W]
Total: _____
1.88[W]

Background Reading

EMG design:
http://www.instructables.com/id/Muscle-EMG-Sensor-for-a-Microcontroller/#step0

EMG backgound information:
http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1455479/pdf/bpo_v8_p11_m115.pdf

NPN transistor as a switch:
techho**use**.brown.edu/~d**morris**/projects/tutorials/**transistor**.**switches**.pdf

EMG Sample Rates:
http://www.biopac.com/sample-rates-for-different-signals

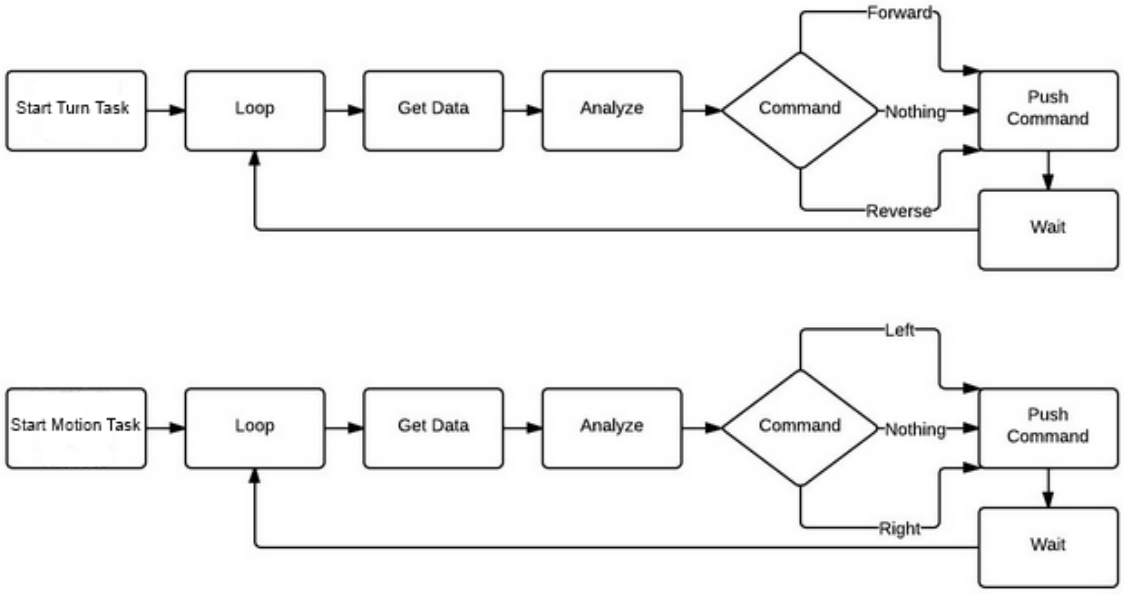
Device Controlled RC Car Capstone Project:
http://www.ece.ualberta.ca/~elliott/ece492/projects/2012w/g1_iOS_RC_Car/G01%20CapstoneProject--FinalReport.pdf

Software Design

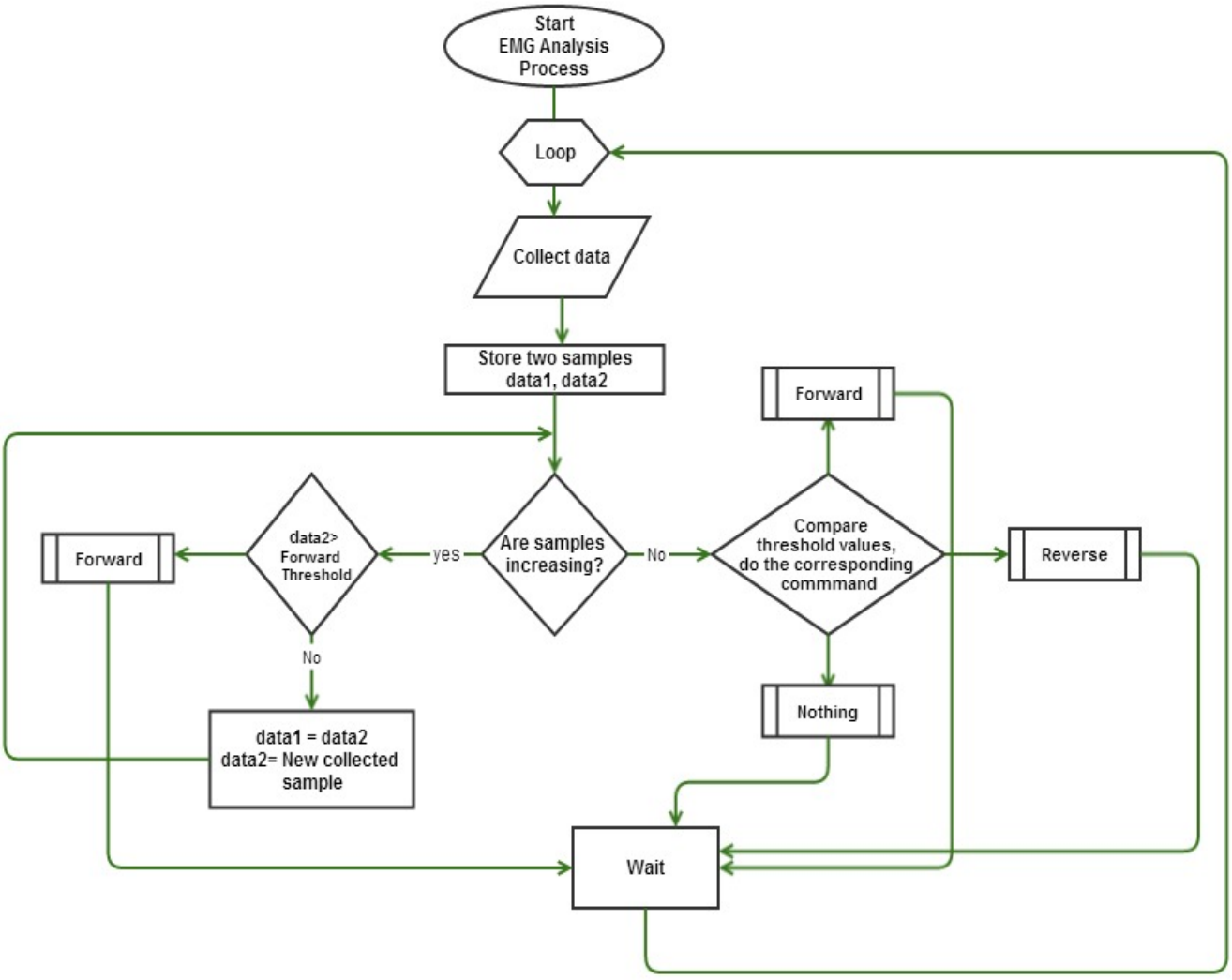
Software was split into two tasks, one for handling turning computation and the other for handling the drive computations. Each task uses the same algorithm to determine the command the user wants to send to the car. The algorithm takes two sequential data samples from the EMG and checks whether they are increasing or not. If the samples are increasing the program continues to analyze samples until a falling edge is detected or the values surpass the forward/right threshold. If the falling edge values are inside the reverse/left region or the values are above the forward/right threshold then the corresponding command will be send to the transmitter. The

algorithm will continue collecting samples but will only check to see if the values drop below the stop threshold. Once the stop threshold has been reached the program sends the stop command to the RC car and then returns to checking for a rising edge.

Top level software diagram



Block Diagram of Algorithm



Test plan

Our test plan was divided into four main groups of independent testing. The first one was checking the correct functionality of DE0 Nano board. Second was testing that our rc car was moving accordingly with respect to the commands sent through the board.Third was for checking that the EMG signals collected and processed properly by the ADC. Last one was inspecting the control algorithm regarding the correct classifications of the signals generated by user.

Hardware Testing

DE0 nano board: Testing the DE0 nano board was fairly easy. DE0-Nano Control Panel allowed us to access various components of the board from a host computer and by using it; we tested sdram, leds, switch, and ADC components of the board. Additionally, we loaded a simple program onto the board, turned it off and turned it back on, and observed that the program was still running implying that flash memory worked.

Hardware-Dependent Software Testing

Connection from GPIO pins to the transmitter: We used four pins of GPIO header where each pin represented one control on the transmitter. Once the bit was set, it was supposed to connect the NPN transistor to 3.3 V logic signal to interact with the 9V transmitter. The problem we encountered in this stage was that GPIO header had different mapping than the original assignments. So, we remapped the necessary pins and were able to successfully control the car through those pins.

Data acquisition of the EMG: Initially we built a system which had four main components: the ADC Controller, a Nios II processor, on-chip memory, and LEDs. EMG was connected to one of the 8 channels of the ADC and the highest 8 bits for that channel were displaced on the LEDs. Although with this system we verified that different motions produced different values and the system was instant to human's reaction, we still could not collect the samples as data for further inspection as the on-chip memory was not enough to include required C library functions. Sdram was added to our system to fix this issue, although it brought up a timing problem caused by the clock skew on the DE0 nano board. It turned out that for proper operation of the SDRAM chip, it was necessary that its clock signal led the Nios II system clock, by 3 nanoseconds. Using a phase-locked loop circuit, we were able to match the requirement and properly collect the data for developing our algorithm.

Algorithm classifications: Lastly, before system integration testing, we checked the classified samples by our algorithm to ensure its correctness. We performed the chosen gestures many times and displayed the encoded signal command on the screen and improved our algorithm up to the point where we were detecting the right commands at all times.

Software

Every-time a task wanted to access the ADC or the Transmitter, we used a semaphore to lock it up and ensure that each task had an exclusive access of those resources. In addition, we powered the system on and left it to run at least for an hour and then proceeded to test the system to confirm that all software components were still functioning properly without any occurrences of deadlocks or slow-downs.

Once we successfully finished unit tests, we integrated all the parts together to see if they still preserved their original functionality.

Results of experiments and characterization:

Due to time constraints, we opted for an adaptive thresholding algorithm, as explained above. There was no calibration since it was used on one user - This was a source of errors since different users do have different thresholds.
A muscle has many threshold values depending on the amount of effort given to it, for our purposes we used three: relaxed, high strength, and medium strength. These provided very clear threshold values that upon close observation were then translated into constant values. While the algorithm was imperfect, it was accurate - it predicted the right command most of the time. However for every new command the user must relax first in order for the algorithm to determine the new command.
There is no clear benchmarking that can be used to determine the algorithm's performance, however, the worst case is when the signal does not match any of the threshold values in which case it stays in the stop mode. Otherwise the algorithm operates as O(1), that is constant, where each command takes the same number of samples to determine the next command.

Safety

Our project had very low risk associated with it. Any risk would come from the connection of the EMGs to the users arm. These risks are very low because these are non invasive methods of monitoring physiological output. The EMG is an electrical recording of muscle activity. The maximum voltage that will be connected to the circuit is 9V. The voltage will be supplied by a 9V battery. There is no danger of electrocution at this voltage. The RC Car that we intend to use had a kill switch built into our software to ensure we can stop the car at any time. It's range is 20m. The car's speed and mass we're created with children in mind since they are the primary users of such a car. So we see no safety concerns related to the RC car itself. For these reasons stated, we believe that our project was of low risk.

Environmental Impact

The hazardous substance that our EMG controlled RC car contains is lead from the solder and is not RoHS compliant. An alternative to this is lead free solder, but unfortunately we do not have access to this. Other parts of our project that have an impact on the environment are the batteries we used to power our transmitter and RC Car. To reduce the impact from these we can recycle the batteries. Another product that has an impact on the environment are the electrodes used to capture the electrical signals from the users' arm. These are disposable, so to reduce the impact on the environment we attempted to limit our use of them, and reused them when possible.

Sustainability

To create a more sustainable project we could utilize solar power in place of our batteries used the the RC car, the transmitter and for the EMG.

Power consumption:

9V batteries: Power Consumption=IV x Time → P = 30mA x 9V x 3600s = 972J

RC Car: Power Consumption = IV x Time → P = 2.5A x 6V x 3600s = 54kJ

DE0 Nano: Power Consumption = IV x Time → P = 500mA x 5V x 3600s = 9kJ

Costs per year:

9V batteries = $0.21/year

RC Car = $11.43/year

DE0 nano = $1.91/year

References

[1] Altera Corporation. (2014, February 28). *DE0-Nano Development Board*. [Online]. Available: http://www.altera.com/products/devkits/partners/kit-terasic-de0-nano.html

[2] B. Kaminsky. (2014, February 28). *DIY Muscle Sensor / EMG Circuit for a Microcontroller*. [Online]. Available: http://www.instructables.com/id/Muscle-EMG-Sensor-for-a-Microcontroller/#step0

[3] MATTEL. (2014, February 28). *Tyco R/C Home Page*. [Online]. Available: http://www.rc-ratz.net/

[4] Altera Corporation. (2014, February 28). *Nios II Processor: The World's Most Versatile Embedded Processor*. [Online]. Available: http://www.altera.com/devices/processor/nios2/ni2-index.html

[5] Altera Corporation. (2014, February 28). *DDR and DDR2 SDRAM High-Performance Controller MegaCore Functions.* [Online]. Available: http://www.altera.com/products/ip/iup/memory/m-alt-hpddr2.html

[6] Altera Corporation. (2014, February 28). *QDRII SRAM Controller MegaCore Function*. [Online]. Available: http://www.altera.com/products/ip/iup/memory/m-alt-qdrii-sram.html

[7] Altera Corporation. (2014, February 28). *SOPC Builder User Guide.* [PDF]. Available: http://www.altera.com/literature/ug/ug_sopc_builder.pdf

[8] D. Morris. (2014, February 28). *Using transistors as switches.* [PDF]. Available: techho**use**.brown.edu/~d**morris**/projects/tutorials/**transistor**.**switches**.pdf

Appendices

Quick start manual

Hardware Setup

1. Replace the INA 106 ICs (we believe this was damaged during the demo)
2. The red banana plugs go to the positive terminals of the power supply, the black banana plugs go to the negative terminals of the power supply.
3. Set the power supply to 3.1V to get correct threshold values. Current is set to approximately 2mA.
4. Connect the green lead to the electrode on the elbow bone. The red lead connects to the electrode placed at the end of the muscle, and the yellow lead connects to the electrode placed in the middle of the muscle.
5. Connect the 26 pin ribbon cable from the EMGs so that the ground pin on the EMG corresponds to the pin 26 on the ADC, and the outputs of the EMG correspond to Channels 0 and 1 on the ADC (pins 24 and 25)
6. Connect the transmitter to the DE0 Nano the ground pin from the transistor switches correspond to the ground pin on the DE0 Nano (pin 12), the switches correspond to pins 2,4,5 and 6 on the DE0 Nano.
7. Connect a 9V battery to the transmitter
8. Turn on RC car

*The first EMG (closest to the banana plugs) controls the forward and reverse commands. The second EMG controls the left and right turn commands.

Software Setup

Volatile

1. Generate Qsys
2. Compile Quartus project
3. Program .SOF file to DE0 nano
4. Create BSP project and link it to the gesture_rover project in Eclipse
5. Run .ELF as NIOSII project

Non-volatile

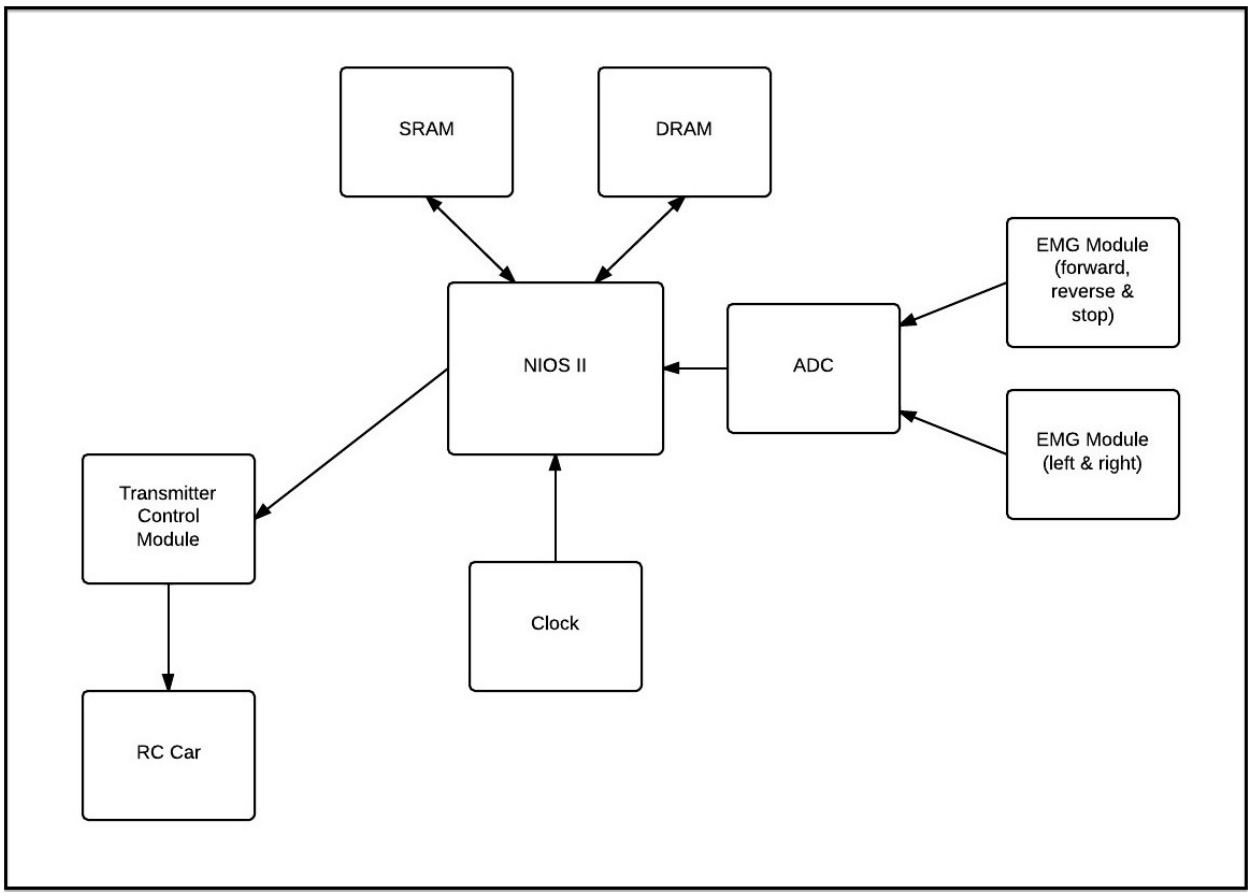
1) Generate Qsys
2) Compile Quartus project
3) Program .SOF file to DE0 nano
4) Create BSP project and link it to the gesture_rover project in Eclipse
5) Use NIOSII Flash Programmer to flash the .POF and .ELF to DE0 nano

Future work

There were many things that wanted to be achieved but due to time constraints we could not pursue. For instance, the original plan was to use a gyroscope to control steering - for this we had a accelerometer/gyroscope (MPU-6050) from SparkFun that would communicate with the DE0 via an OpenCores I2C module - however, this was not accomplished.
This project has a lot of potential - If future work on this project is possible improvements on the following would be pursued:

- Improve robustness of the EMGs - This is possible by making a PCB off the circuit we used, and including measures to lower interference from outside noise into the circuit.
- Make the EMGs wireless such that there is no wires between the microprocessor and the EMGs.
- Use better algorithms (i.e: fourier analysis algorithms) and machine learning to better determine commands of more than one user.

Hardware Documentation



TL072 IC: http://www.ti.com/lit/ds/symlink/tl072.pdf
INA106 IC: http://www.ti.com/lit/ds/symlink/tl072.pdf
IC-MPU-6050:
http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Components/General%20IC/PS-MPU-6000A.pdf
DE0-nano:
http://www.altera.com/education/univ/materials/boards/de0-nano/unv-de0-nano-board.html

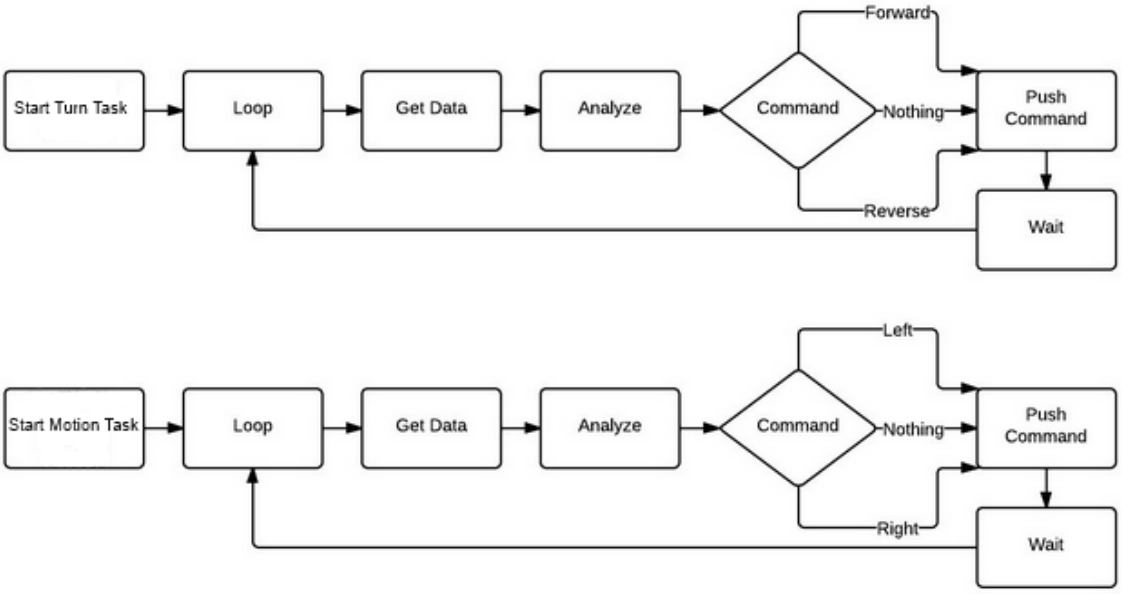
Source Code

Complete source code can be found on our Github repository:
https://github.com/grantdhunter/ECE492

Projects:
emg_test (**T**) - project used to test the software in charge of reading the EMG data from the ADC
imu_test (**C**) - project used to test the software that was in charge of communicating with the IMU (was not used in the final project)
transmitter_test (**T**) - project used to test the software in charge of sending data to the transmitter

gesture_rover(**T**) - main project that contains all subsystems integrated together
- main.cpp(**T**) - contains the main algorithm that determines what command the user is sending
- emgInterface.cpp/h(**T**) - EMG object that reads from the ADC
- transmitterInterface.cpp/h(**T**) - transmitter object that send the commands to the transmitter
- StatusLED.cpp/h(**T**) - LED status lights object that displays which command has been sent
- dataType.h(**T**) - header file containing data types, contains most of stdint.h. It was used because of strange linking errors in Eclipse.

Top level software diagram



Block Diagram of Algorithm