0db3d1c7-0f92-49f2-85a9-f425f6553da4

-store strings in a vector

-Convert vector of strings to ints

-Convert to voltage (float)

-Convert to PSI and Newtons

-Printf

ROS Development Subteam

● Develop a C++ code that will perform measurement unit conversion for each sensor

reading stored in the received message from the Serial port:

○ Sent Message = “ADC\r”

○ Received Message = “char,char,char,...,char\r” with: 1011, 678,187, 98, 1020, 345, 657, 976, 235, 275, 376, 53\r

■ number of arguments = 12

■ argument char = 0~1023 (positive integers)

■ **arguments 0-3 = FSR measurement**

FSR: <https://www.sparkfun.com/products/9376>

Coding FSR: <https://learn.adafruit.com/force-sensitive-resistor-fsr/using-an-fsr>

■ **arguments 4-7 = Gauge measurement**

Pressure Gauge:

[https://www.convertunits.com/from/psi/to/pound/square+inch+[gauge](https://www.convertunits.com/from/psi/to/pound/square+inch+%5Bgauge)] (New)

<https://www.digikey.com/products/en?keywords=ASDXAVX100PGAA5>

■ **arguments 8-11 = EMG measurement**

Muscle Sensor: <https://www.sparkfun.com/products/13723>

○ Converted unit should be:

■ FSR = Newtons | Gauge = PSI | EMG = Voltage

* <https://arduino.stackexchange.com/questions/39991/reading-force-from-an-fsr-in-newtons/39997>
* <https://learn.adafruit.com/force-sensitive-resistor-fsr/using-an-fsr#in-depth-code-for-analog-fsr-measurements>

(Deadline: 02/07/18)

#include <stdio.h> /\* Standard input/output definitions \*/  
#include <unistd.h> /\* UNIX standard function definitions \*/  
#include <fcntl.h> /\* File control definitions \*/  
#include <errno.h> /\* Error number definitions \*/  
#include <termios.h> /\* POSIX terminal control definitions \*/

int FSRFunction(int a){

int result = 0;

result = “equation here”;

return result;

}

};

int GuageFunction(int b){

int result = 0;

result = “Equation here”;

return result;

}

};

int EMGFunction(int c){

int result = 0;

result = “Equation here”;

return result;

}

};

int open\_port(void){  
 int fd; // File descriptor for the serial port  
 fd = open("/dev/pts/1", O\_RDWR | O\_NOCTTY | O\_NDELAY);

if (fd == -1){  
 perror("\nopen\_port: Unable to open serial port - ");  
 }  
 else{  
 printf("\nPort opened successfully!\n");  
 }  
 return (fd);  
}  
  
void set\_port(int fd){  
 struct termios options; // Creates the structure  
 tcgetattr(fd, &options); // Gets the current attributes  
  
 cfsetispeed(&options,B9600); // Sets read speed to 9600 bps  
 cfsetospeed(&options,B9600); // Sets write Speed to 9600 bps  
  
 options.c\_cflag &= ~PARENB; // Disables parity  
 options.c\_cflag &= ~CSTOPB; // Sets the stop bits amount to 1  
 options.c\_cflag &= ~CSIZE; // Clears the mask for setting the data size  
 options.c\_cflag |= CS8; // Set the data bits amount to 8  
 options.c\_cflag &= ~CRTSCTS; // Disables hardware flow control  
 options.c\_cflag |= (CREAD | CLOCAL); // Enables receiver and sets local mode  
 options.c\_iflag &= ~(IXON | IXOFF | IXANY); // Disables software flow control  
 options.c\_iflag &= ~(ICANON | ECHO | ECHOE | ISIG); // Selects raw input mode  
 options.c\_oflag &= ~OPOST; // Selects raw output mode  
  
 if((tcsetattr(fd, TCSANOW, &options)) != 0){ // Sets the new attributes  
 perror("\nset\_port: Error in setting attributes - ");  
 }  
 else{  
 printf("\n| Baud rate = 9600 bps | Stop bits = 1 | Parity = None |\n");  
 }  
}

int write\_port(int fd, char \*buffer\_write, size\_t buffer\_size){  
 int bytes\_write = write(fd, buffer\_write, buffer\_size);  
 if (bytes\_write < 0){  
 fputs("write() of bytes failed!\n", stderr);  
 }  
 return (bytes\_write);  
}  
  
int read\_port(int fd, char \*buffer\_read, size\_t buffer\_size){  
 int bytes\_read = read(fd, buffer\_read, buffer\_size);  
 return (bytes\_read);  
}  
  
void close\_port(int fd){  
 close(fd);

}

int main(){  
 int fd, bytes\_write = 0, bytes\_read = 0;  
 char buffer\_write[] = "What is your name?:";  
 char buffer\_read[32];  
  
 fd = open\_port(); // Function to open serial port  
 set\_port(fd); // Function to set serial port options

while(bytes\_read < 1){ // Reads until at least 1 byte is received  
 bytes\_read = read\_port(fd, buffer\_read, sizeof(buffer\_read)); // Function to read from serial port  
 }

close\_port(fd); // Function to close serial port  
 return 0;

}