**Guide to the STARX Exoskeleton Code**

Table of Contents

[Introduction 2](#_Toc519264605)

[Libraries Used 3](#_Toc519264606)

[serial.h Libraries: 3](#_Toc519264607)

[arduinomicro.h Libraries 4](#_Toc519264608)

[converter.h Libraries 4](#_Toc519264609)

[pid.h Libraries 5](#_Toc519264610)

[.cpp Files 5](#_Toc519264611)

[serial.cpp 5](#_Toc519264612)

[arduinomicro.cpp 6](#_Toc519264613)

[converter.cpp 6](#_Toc519264614)

[pid.cpp 6](#_Toc519264615)

[exosuit.cpp 7](#_Toc519264616)

[main.cpp 7](#_Toc519264617)

# Introduction



Top begin in figure 1 is a flowchart of the STARX exoskeleton code as of version 2.2, but until a Robot Operating System in implemented we will be just using C++ coding. To talk about figure 1 a bit the code ends at the main.cpp function. From there it pulls data gathered by the exosuit.h function

****To start out I better explain a few beginning things that deals with programming. First of all C++ programming is a language that is very logical and requires linear thinking. So, when we start creating a program for C++ we must have a “main” function to start off. In figure 2 is our main function as of version 2.2. that calls the exosuit.h header; only the main function in our code needs to have a “main” the rest are called to be used in the code.

Figure : Flowchart for STARX Exoskeleton code

**Header** files serve two purposes:

* System header files declare the interfaces to parts of the operating system. You include them in your program to supply the definitions and declarations you need to invoke system calls and libraries.

Figure : This is the main function that call all other functions to fun the program for the exoskeleton.

* Your own header files contain declarations for interfaces between the source files of your program. Each time you have a group of related declarations and macro definitions all or most of which are needed in several different source files, it is a good idea to create a header file for them.
* Headers could be inside the .cpp code they are called to, but to make the code more clear and less error prone a header is created to make sure the declarations and libraries needed are already available.

# Libraries Used

## serial.h Libraries:

#include <stdio.h>

This allows for the input/output library to be used. Uses streams to operate with physical devices such as keyboards, printers, Terminals, etc. <http://www.cplusplus.com/reference/cstdio/>

#include <unistd.h>

Allows the system to be a POSIX operating system API. POSIX is portable operating system interface that is a family of standards by IEEE to maintain compatibility between operating systems to clarify and make uniform the application programming interface. We can now port to MacOSX for example because of this.

#include <fcntl.h>

This header shall define the following symbolic constants for the command argument used by fcntl(). The values shall be unique and shall be suitable for use in #if preprocessing directives. To be clear about what this does it just adds more commands that <unistd.h> does not add. To find exact commands and applications this adds go to: <https://www.mankier.com/0p/fcntl.h>

#include <errno.h>

This library defines macros for reporting and retrieving error conditions through error codes. Because of this library the systems can indicate, in the event of an error, what when wrong. To go into more detail go to: <https://www.tutorialspoint.com/c_standard_library/errorno_h.htm>

#include <termios.h>

This is a library that provides the terminal interface for the POSIX compatibility. A good reference to explain further: <http://manpages.ubuntu.com/manpages/trusty/man7/termios.h.7posix.html>

#include <string.h>

This library defines one variable type, one macro, and various functions for manipulating arrays of characters. To find what macros and variables it includes for use go to: <https://www.tutorialspoint.com/c_standard_library/string_h.htm>

## arduinomicro.h Libraries

#include <string.h>

This library defines one variable type, one macro, and various functions for manipulating arrays of characters. To find what macros and variables it includes for use go to: <https://www.tutorialspoint.com/c_standard_library/string_h.htm>

#include <vector>

Vectors are same as dynamic arrays with the ability to resize itself automatically when an element is inserted or deleted, with their storage being handled automatically by the container.

<https://www.geeksforgeeks.org/vector-in-cpp-stl/>

## converter.h Libraries

#include <cmath>

This library dclares a set of functions to compute common mathematical operations and transformations. <http://www.cplusplus.com/reference/cmath/>

#include <iostream>

C++ standard library for input and output arduments.

<https://en.wikipedia.org/wiki/Input/output_(C%2B%2B)#Input/output_streams>

#include <sstream>

Sometimes it is very convenient to use stringstream to convert between strings and other numerical types. The usage of stringstream is similar to the usage of iostream. Stringstreams can be used to both read strings and write data into strings.

The basic member functions of stringstream class are:

str(), which returns the contents of its buffer in string type.

str(string), which set the contents of the buffer to the string argument.

<https://stackoverflow.com/questions/20594520/what-exactly-does-stringstream-do>

#include <string.h>

This library defines one variable type, one macro, and various functions for manipulating arrays of characters. To find what macros and variables it includes for use go to: <https://www.tutorialspoint.com/c_standard_library/string_h.htm>

#include <vector>

Vectors are same as dynamic arrays with the ability to resize itself automatically when an element is inserted or deleted, with their storage being handled automatically by the container.

<https://www.geeksforgeeks.org/vector-in-cpp-stl/>

## pid.h Libraries

#include <stdio.h>

This allows for the input/output library to be used. Uses streams to operate with physical devices such as keyboards, printers, Terminals, etc. <http://www.cplusplus.com/reference/cstdio/>

#include <sys/time.h>

This library allows the use of date and time utilities within the code to be used

<https://www.quora.com/What-is-the-main-use-of-including-the-header-file-sys-time-h-Are-time-h-and-sys-time-h-the-same>

# .cpp Files

## serial.cpp



class SerialPort {}

creates a class called SerialPort.

Public:

This allows this file to be used in other functions that is stated after the class initialization.

speed\_t

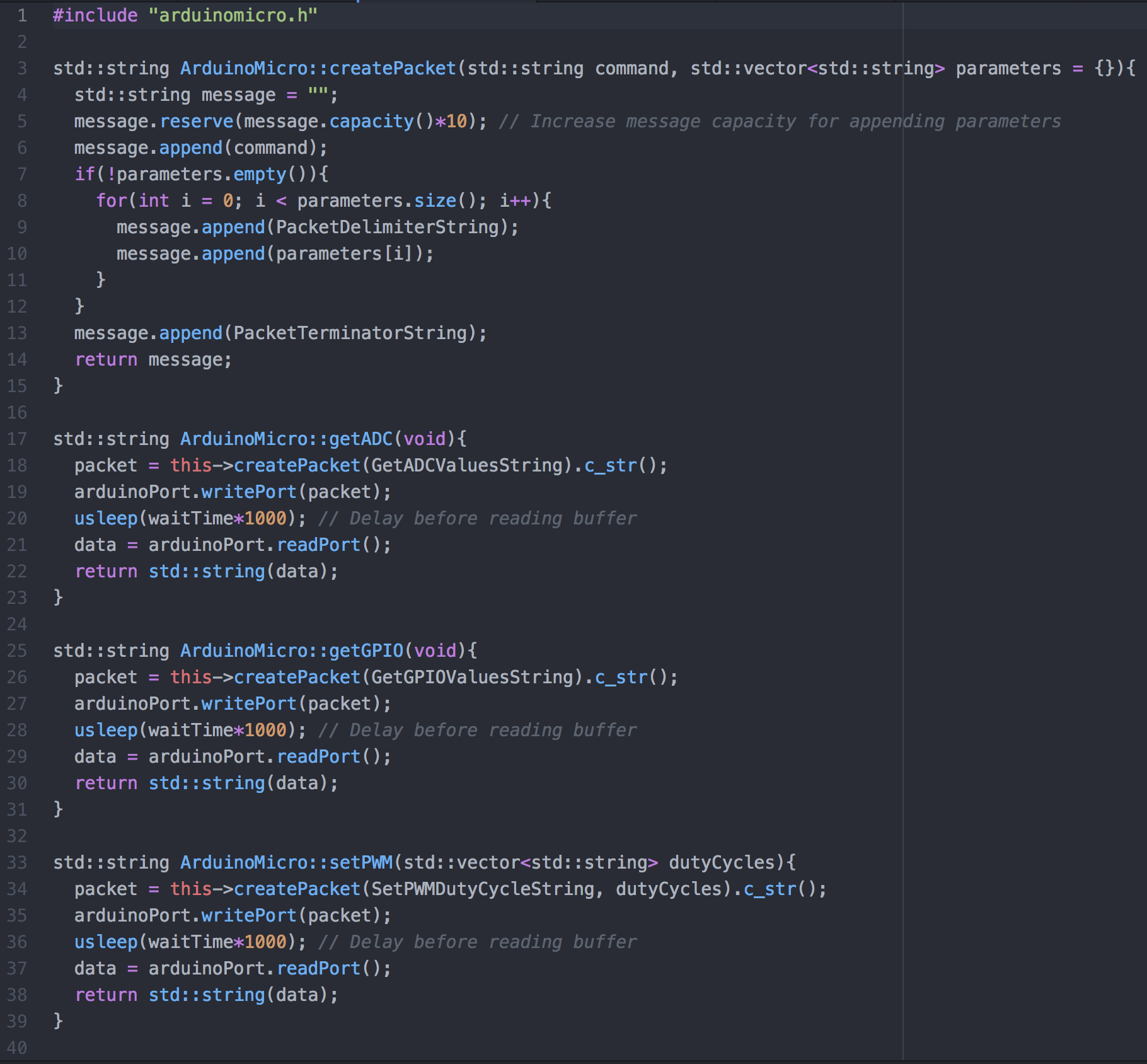
This is something from the termios.h library that is used for terminal baud rates (rate at which information is transferred, eg. Serial port is 9600 bits per second or 9600 baud)

constBaud

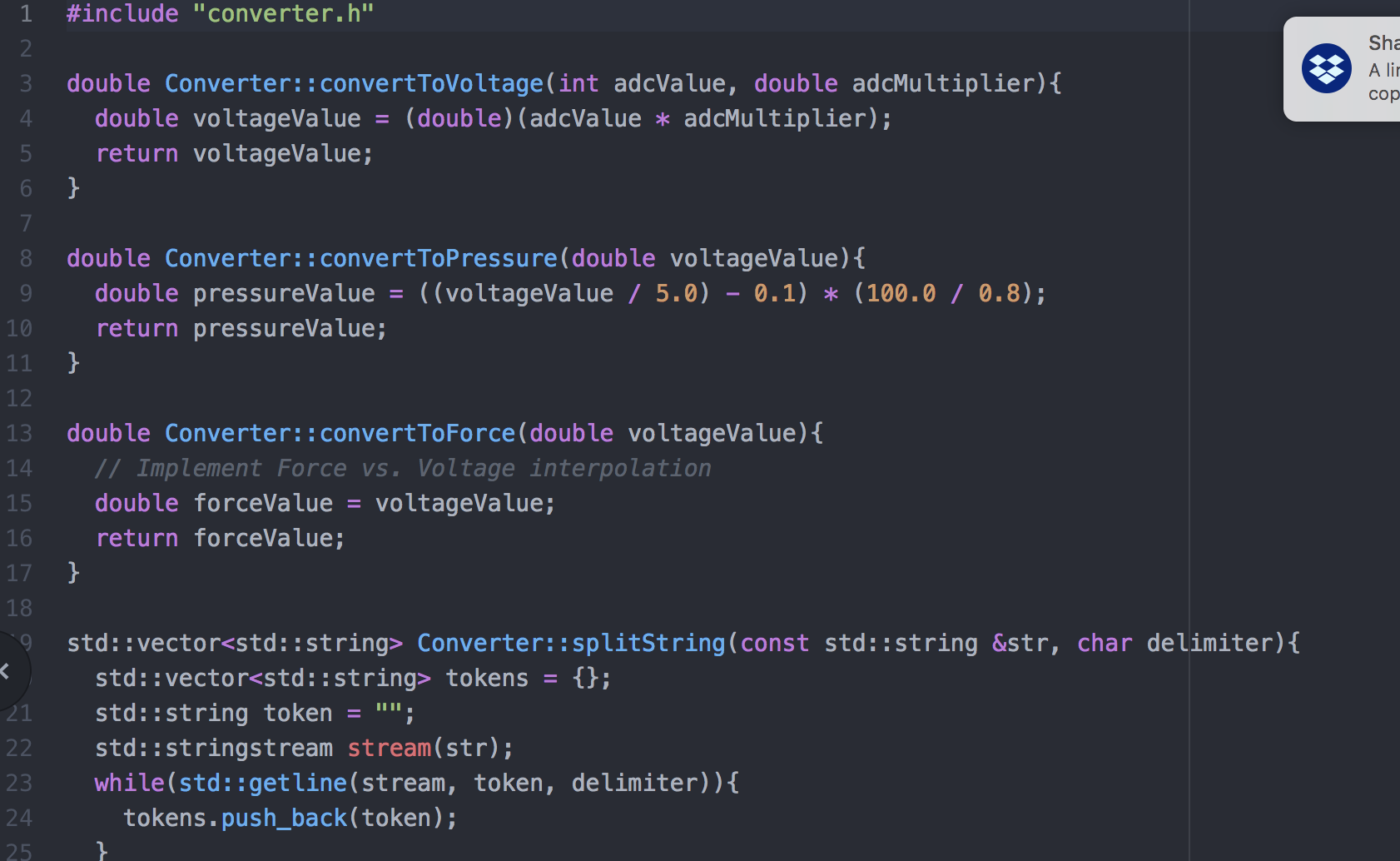
This is right after speed\_t function and what it does is holds a place for a value, in this case a bandwidth for the serial port. From the header file where this variable is initialized it is then utilized within the serial.cpp file where the computer runs this variable through a switch case function to find the correct band rate that is used, and this variable then holds that value for the baud rate.

After the switch case the code goes onto setting up how the code is transferred in and out of the computer. ***I can add a section about this later.***

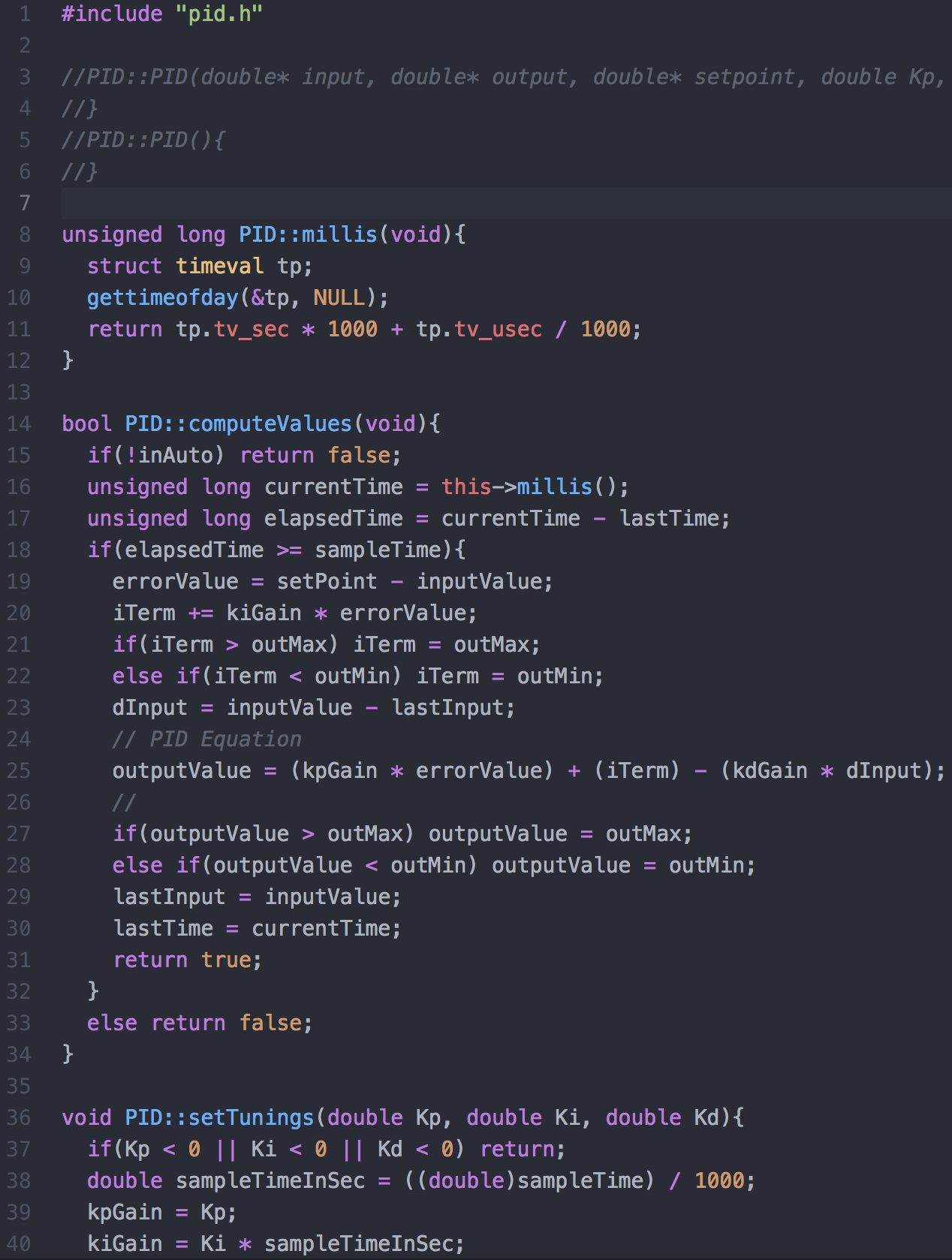
## arduinomicro.cpp

******c

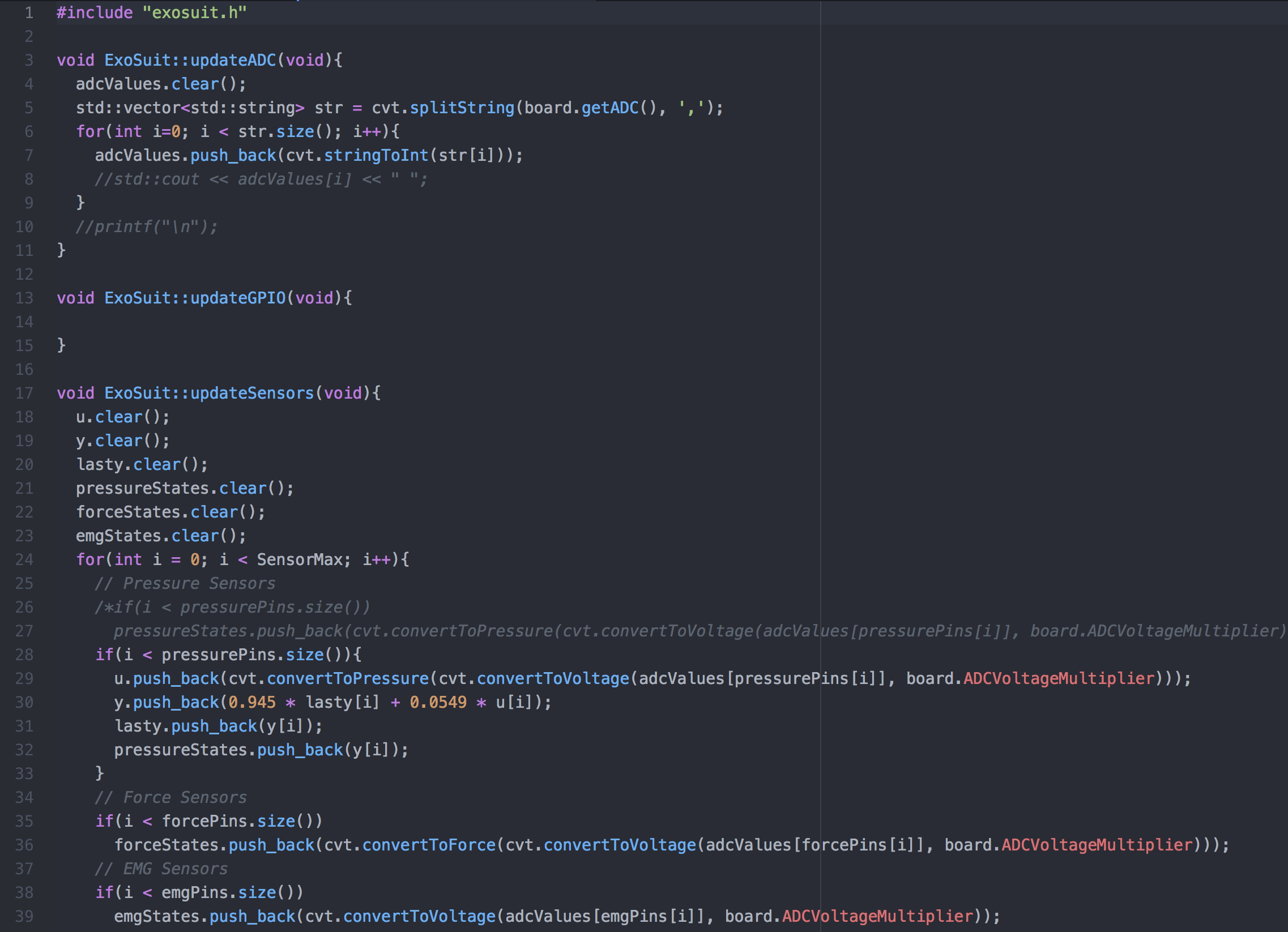
## converter.cpp

***Again I am going to wait to explain things till later in this section.***

## pid.cpp

***I am going to wait to explain*** ***here***

## exosuit.cpp



## main.cpp

***I am going to wait to explain things here.***