# EMBEDDED SYSTEMS REPORT

Lab Group: G4a

UCARD: 160159871

Word Count: 2336



Electronic & Electrical Engineering
The University of Sheffield

**Introduction**

This lab investigates the use of LabVIEW, a graphical programming language and development environment to interface with external devices such as a servo control system. Simple programs will be created and debugged to perform certain tasks, and there will be an introduction to performing data acquisition within a computer system. Furthermore, this lab looks at working with embedded systems using the myRIO board by National Instruments (NI), which will be programmed using LabView.

**Background**

The LabView graphical programming language is strictly typed, meaning there is no code to physically type, hence a program is made by connecting a network of modules, in the form of icons. The benefit of creating a program this way compared to physically programming it comes down to reliability, speed and compatibility. For instance, a typed program could be made from many sections of code written by different entities, to different specifications. There may be no guarantee that these sections of code will work together, which leads to time consuming debugging and testing. In contrast, the modules within LabView are pre-written, pre-tested and are compatible with each other, resulting in a program that is simpler to create and more robust. A different example of a graphical programming application is Simulink, where a program is written by icons rather than lines of code.

An industry application of this could be air-bag systems for cars, where rather than each manufacturer creating their own code, they import a pre-written, heavily tested program that will be compatible and is guaranteed to operate reliably.

Data acquisition is the process of measuring physical quantities, such as temperature or a voltage, and converting them into a form which can be understood by a computer program, such as an electronic signal, to be stored or displayed. This process is used in virtually every area of science and engineering, from complex control systems to a simple volt meter. [1]

In the context of this lab, the data acquisition process is used as part of the control system that controls the servo motor. Electrical signals from its potentiometer and tachometer are converted into a suitable data type to determine and display the servos angular position speed, which can then be used as feedback to accurately control the operation of the servo.

An embedded system can be part of a larger electrical/mechanical system, and has a specific dedicated function that it performs. They have many forms, from being small discreet microprocessor systems which perform a single task, to complex systems that perform multiple tasks with user interfaces. An Arduino is a classic example if an embedded system, where a microprocessor (ATMega328) can be used as an I/O device to control external motors, lights etc. [2]

For example, within a cars electronic controls system, and embedded system could be used to perform a single, dedicated function, such as operating the seatbelt pre-tensioner in the event of a crash. This embedded system would be programmed once, and would operate reliably because it has no other functions to control and will not be affected by software updates in other systems. If another system was to fail, the embedded system would not be affected.

**Theory**

Hardware Interfaces:

Embedded systems are able to control external mechanical and electrical devices because they have a range of input/output (I/O) ports that can be used to receive and transmit both analogue and digital signals. An example of the I/O interface from an Arduino can be seen below in *figure 1*.

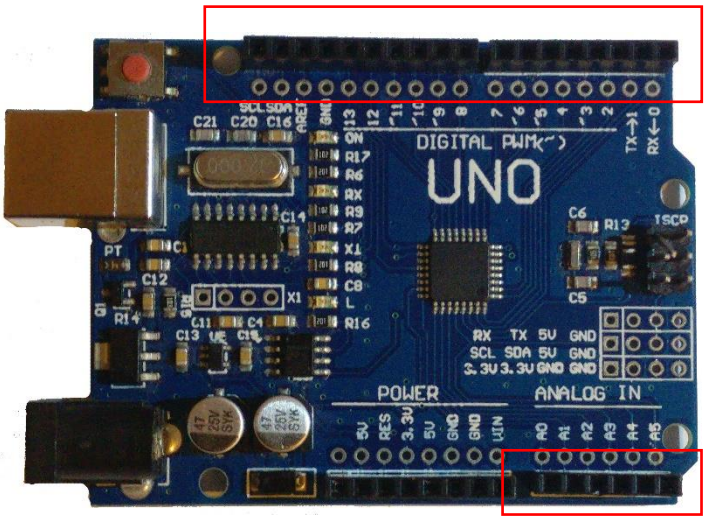


*Figure 1: Arduino I/O Ports.*

However, the myRIO used in the experiment is slightly different to the Arduino as it has a Field Programmable Gate Array (FPGA). The microprocessor in the Arduino is a fixed circuit with specific instructions that need to be used to control it, whereas the FPGA in the myRIO can be configured entirely from scratch to be set up however the programmer decides using low-level Hardware Description Language (HDL). [3]

These I/O pins can work with signals such as PWM, AC analogue signals and serial data, which makes them perfect for many kinds of control systems where data from the device is it controlling can used as an input to the embedded system, to be processed and make changes to the output.

In the context of the lab, the I/O pins from the myRIO are used to send data to the servo, and receive real-time data from the servos potentiometer and tachometer to enable accurate control over its speed and position.

Data Acquisition:

The process of data acquisition (DAQ in LabView), involves taking a signal from a sensor, which could be in the form of a voltage, current or resistance, and converting it into a suitable electronic signal that can be interpreted by a computer, seen in *figure 2*.


*Figure 2: DAQ Process.*

The DAQ device is the middle man between the sensor and computer system. It acts to convert the raw sensor signal into a safe and readable electronic signal through processes such as signal conditioning and analogue to digital conversion (ADC). Signal conditioning involves manipulating a signal that's too noisy or dangerous to directly measure, through processes such as attenuation, amplification and filtering to create a signal suitable for the ADC. The ADC convert this analogue signal into a digital form that can be transferred to the computer to be decoded and analysed. [4]

**Method & Results**

Servo Control:

The first experiment involved using LabView and the ELVIS board as an I/O interface and data acquisition device to control a servo motor as seen in *figure 3*.

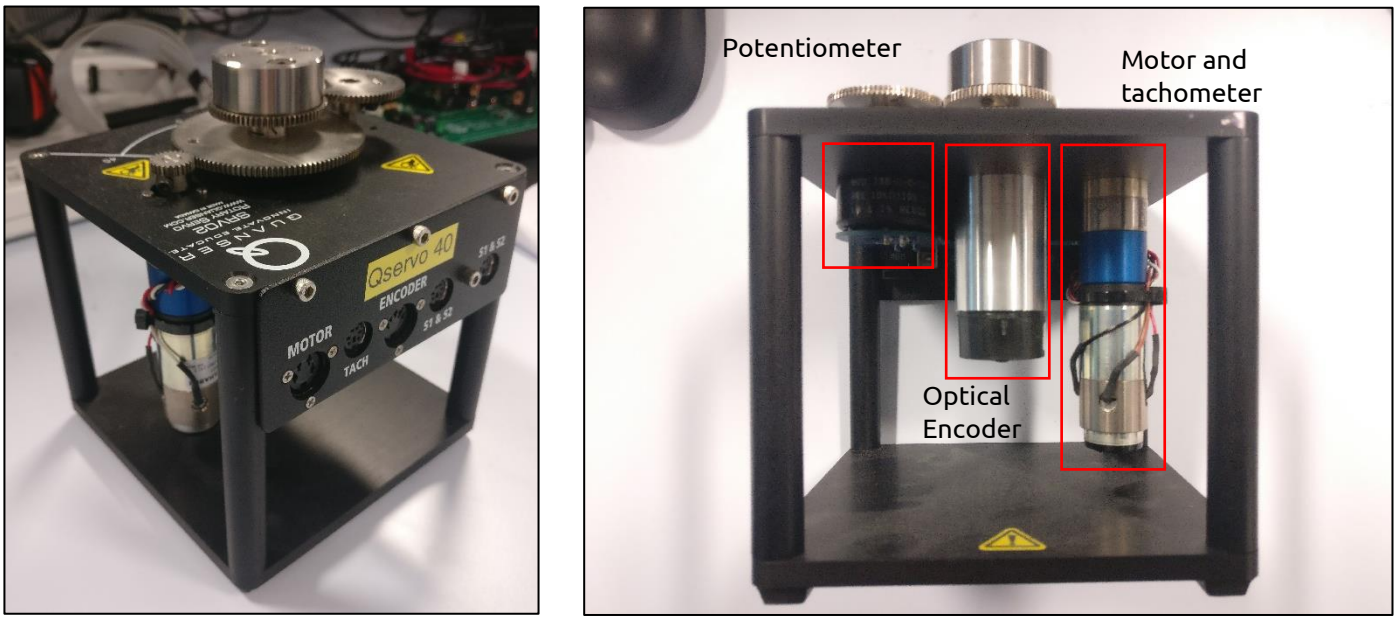


*Figure 3: Servo motor.*

As part of the data acquisition process, the potentiometer and tachometer sensors will provide signals to indicate the servos position and angular speed respectively.

To communicate with the ELVIS board, create a LabView block diagram such as the one in *figure 4*, that can take the data from the sensors and display them accordingly, while manually controlling the motors speed.

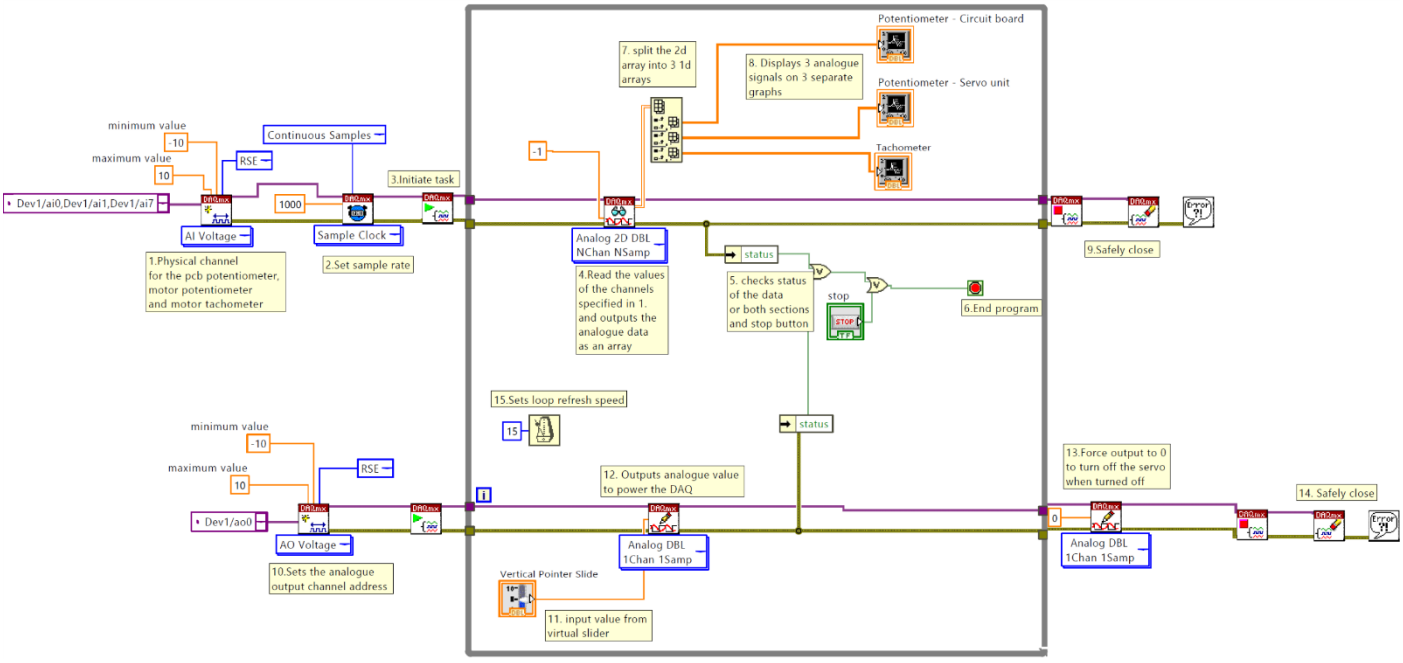


*Figure 4: Servo control block diagram.*

Connect the servo motor to the motor driver unit, and connect the data cables from the servo to the ELVIS board so that LabView can communicate with it.

Run the program, and using the front panel as shown in *figure 5*, the speed of the motor can be controlled using the slider, with the sensor data being shown in real time on the three graphs.

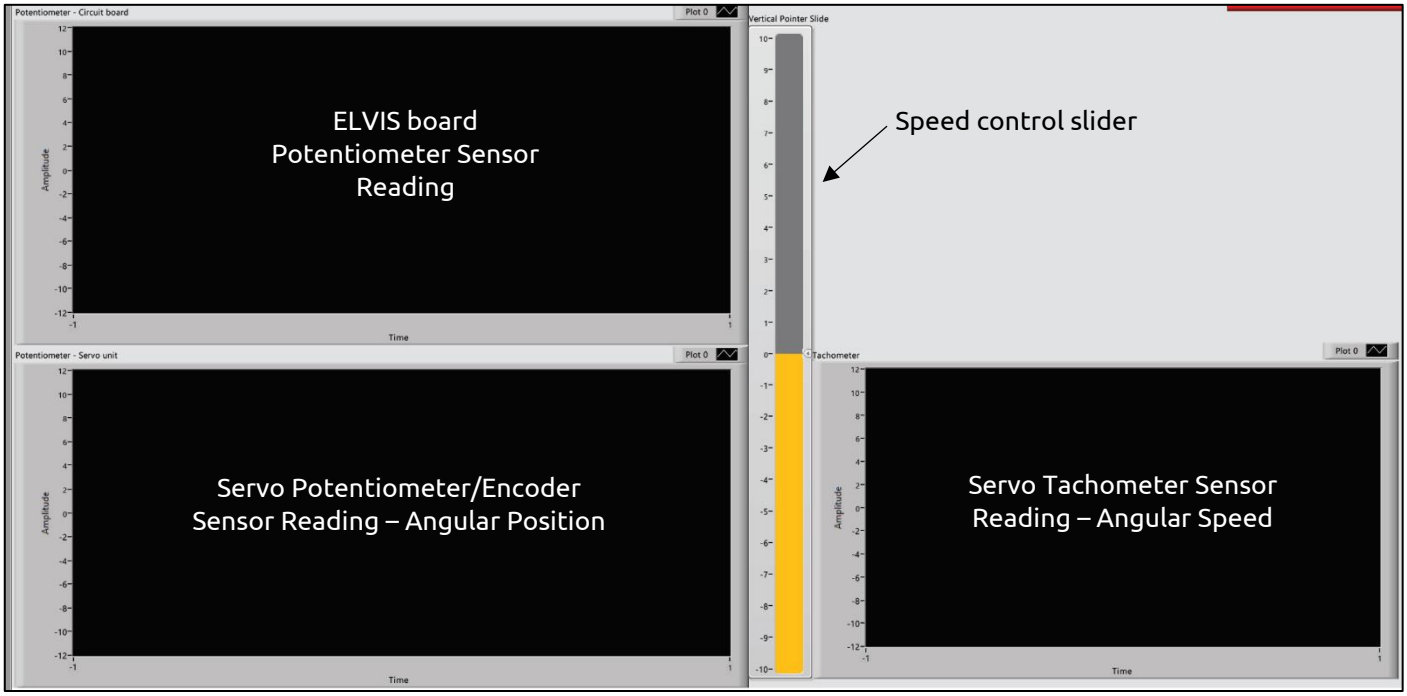


*Figure 5: Servo control front panel.*

The potentiometer/Encoder screen will display a saw-tooth waveform, with each period of the waveform representing one revolution of the servo load. Furthermore, the tachometer will display a DC voltage proportional to the motors speed.

The second experiment involved programming the myRIO board as an embedded system to control LED's.

An example circuit is shown below in *figure 6*.

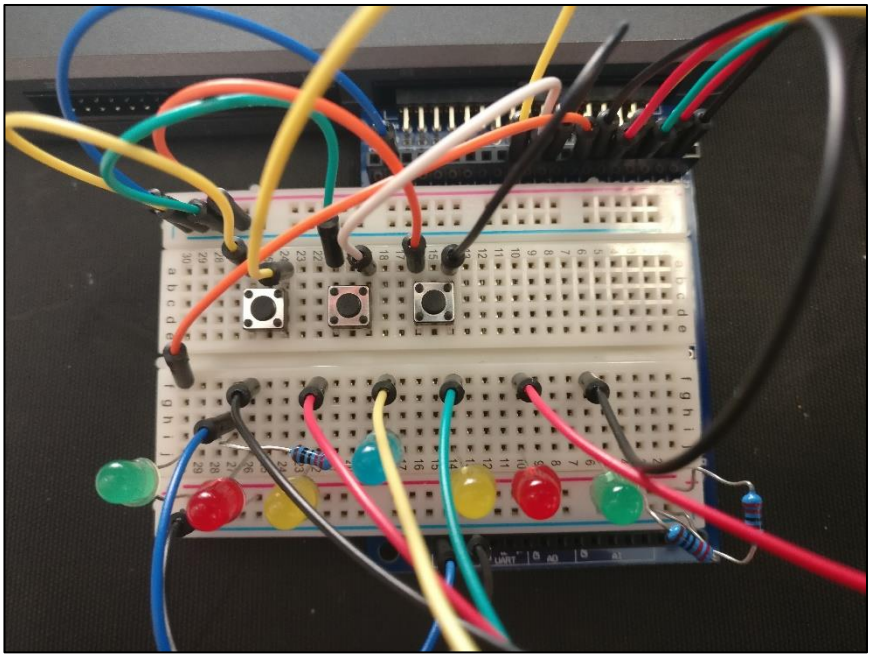

*Figure 6: LED circuit.*

Using LabView, design a block diagram that uses the various I/O pins on the myRIO to input data from switches and potentiometers etc.

An example circuit that causes a 'chasing lights' effect using a series of LED's and buttons seen in *figure 6* is shown below in *figure 7*.

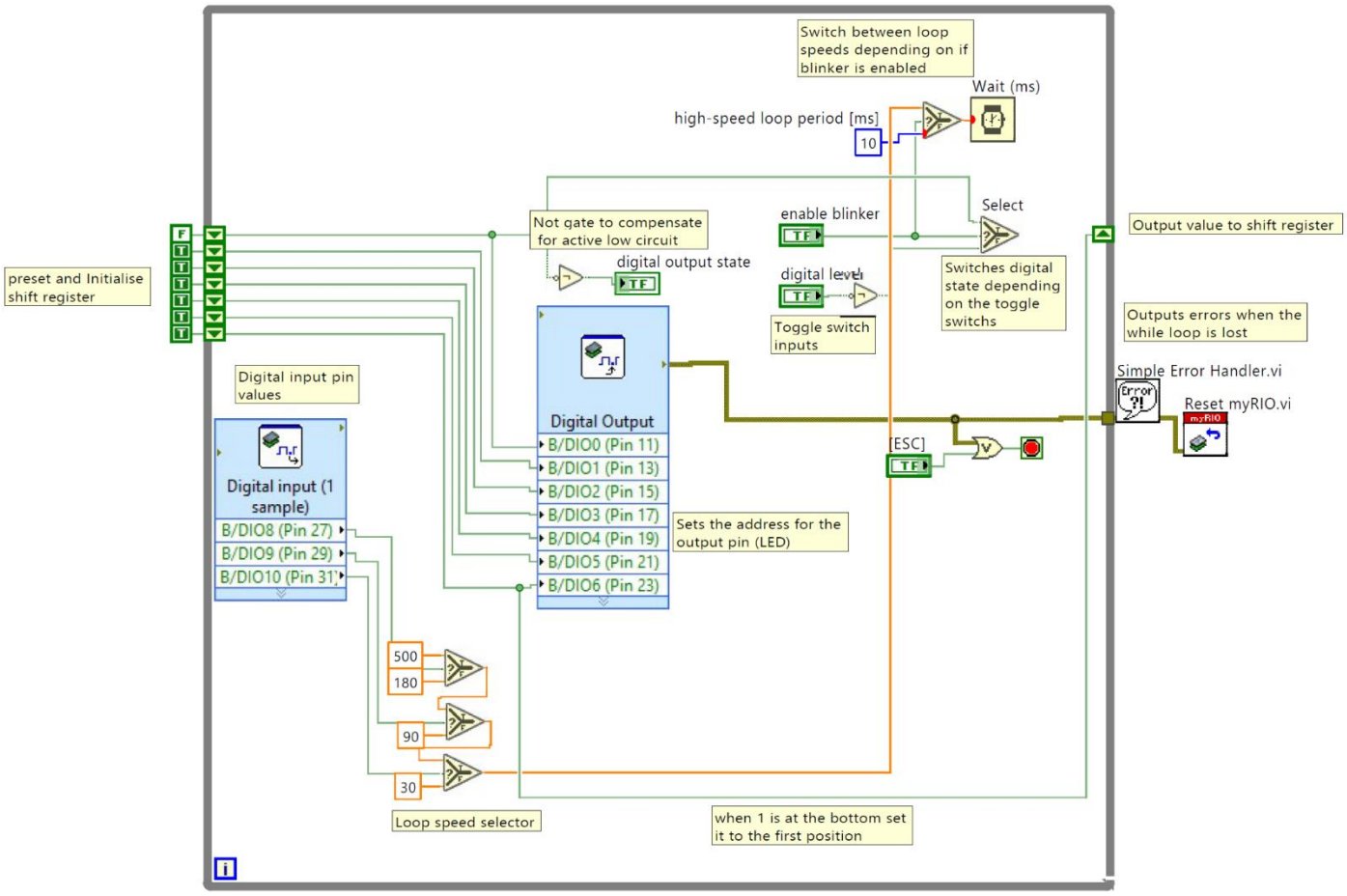


*Figure 7: LED control block diagram.*

Construct the circuit using LED's and switches, making sure to use the correct I/O pins on the myRIO.

Upload the code, and confirm that the circuit behaves as it was programmed.

As expected, once programmed, the myRIO effectively became its own computer, and no longer needed to communicate with LabView for the interfacing to work. When the buttons were pressed, the LED patterns configured worked as intended.

**Discussion**

The first experiment operated as expected, with the software controls in LabView successfully controlling the motor, and the data from the sensors being displayed on the front panel. The ELVIS board acted as the interface between the computer running LabView and the motor driver. The data acquisition process from the block diagram in *figure 4* to collect data from the sensors is shown below in *figure 8*.

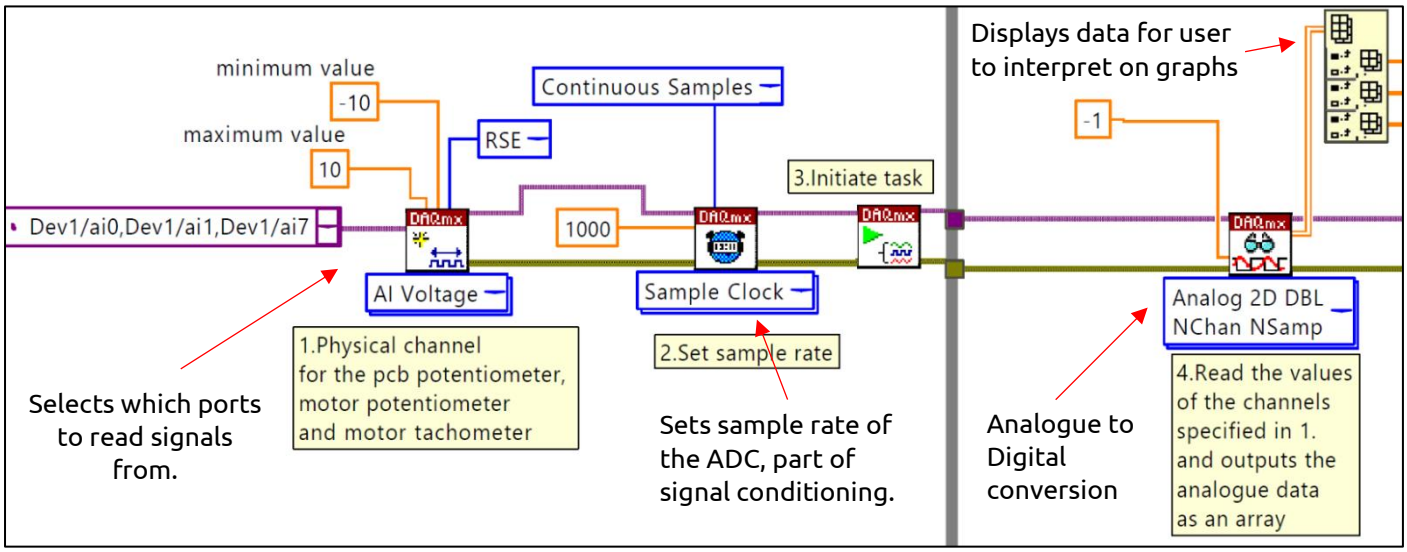


*Figure 7: Servo control DAQ process.*

Is it clear using the graphical programming language how the data is received, conditioned and converted, then displayed.

The output section of the code is shown below in *figure 8*.

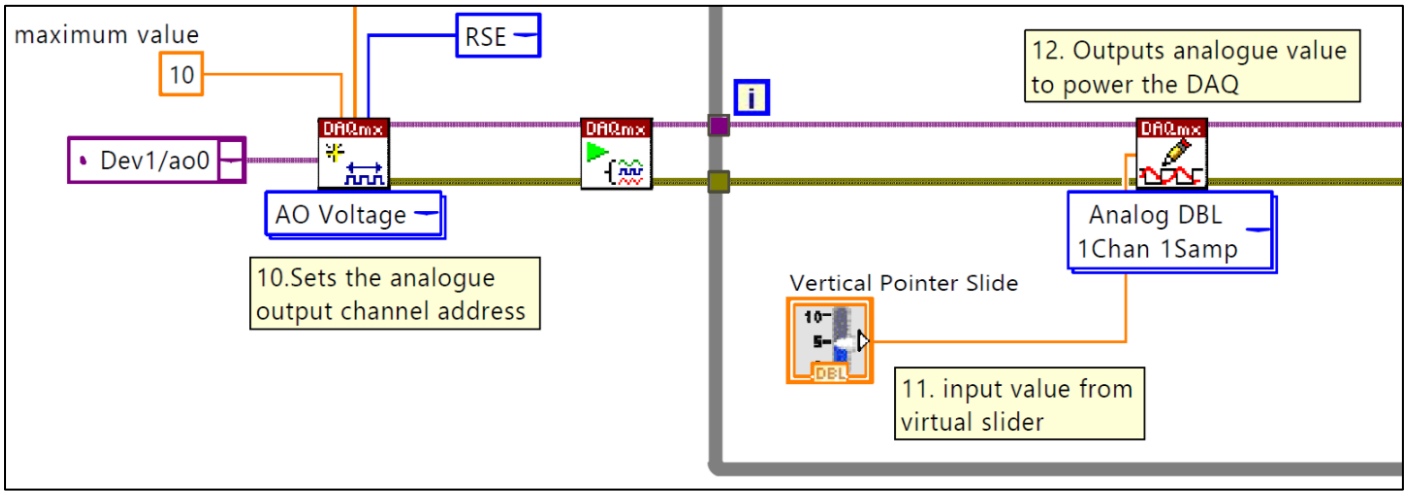


*Figure 8: Servo control output.*

This section of the code takes the signal from the software slider, and outputs a signal to the ELVIS board to control the motor.

The code that controls the LEDs was very similar as seen in *figure 7*, where the inputs from the hardware buttons are received as digital signals, and are then processed to trigger a change in the output.

There are many comparisons to be made between the ELVIS board and the myRIO as an I/O control device, and each have their own specific advantages and disadvantages within different contexts.

Firstly, the ELVIS board in this experiment is merely an interface, used as a data acquisition device to take data from the motor and give it to the computer, relying on communication with LabView to operate. If the ELVIS was to be disconnected from the LabView software, it would no longer function as programmed, and would be unable to interpret data from the motors sensors to make adjustments. This is because the ELVIS does not have its own memory or microprocessor – it relies on the computer running LabView to do the necessary processing.

In contrast, the myRIO is an embedded system, with its own processor, FPGA, memory and I/O ports. Once programmed, its effectively becomes its own computer that can process input data and control its outputs. This enables the myRIO to continue working as it was programmed to do while being disconnected from LabView.

This is a significant difference between the two systems because it makes the myRIO systems and embedded systems as a whole more versatile. For example, the board used on the ELVIS system was custom made to perform a specific task, which is only compatible with the servo motor and cannot be easily modified to be compatible with other devices. However, an embedded system like the myRIO doesn't have a fixed way of operating, and can be programmed in virtually any way to utilise it's available I/O ports. This makes it particularly useful in the real world for prototyping control systems, where it would be unnecessarily expensive to produce a new module for the ELVIS for each iteration of the system design, whereas the myRIO can simply be reprogrammed, and used for many different applications.

Furthermore, in certain aspects, the myRIO has an advantage over the ELVIS system because of its portability. An example of where this is relevant is robotics, where the compactness and independence of the myRIO allows it to be easily connected to a robot, whereas the ELVIS is designed for work bench use and wouldn't operate without being connected to LabView.

However, the rich functionality of the ELVIS system makes it more suitable for use in education and accurate measurements, since it has the capabilities of an oscilloscope, function generator, power supply and millimetre built into one package that is software controlled. This makes it more cost effective compared to buying all the above systems separately, and it has a more versatile way of displaying data accurately within software. The modular design of the ELVIS system with interchangeable PCB modules also makes it perfect for education since there can be dedicated circuits for different experiments.

Another disadvantage of the myRIO is its high price, which limits it availability to many people, whereas a cheap Arduino is both easier to use, can be programmed many different ways and can match almost all the functionality of the myRIO with the right hardware. However, the Arduino is an embedded system designed for hobbyists, whereas the myRIO is based upon the cRIO, which is an industry standard, and hence will be more likely to be used in the real world.

**Conclusion**

Overall, the lab successfully met the initial aims by investigating the use of graphical programming to reliably operate control systems, while exploring the use of embedded systems. For example, through using LabView, it was clear how the data acquisition process was used to collect signals from the servo or switches, to the be processed and influence the output. Furthermore, an understanding of the fundamental differences between the ELVIS system and the myRIO was established, and the strengths that each system has over each other in certain contexts. For example, due to the myRIO's portability and versatility, it is most suitable for control systems, where its range of sensors, I/O and processing power make it perfect for robotics or data collection in the field. In contrast, it makes perfect sense to use the ELVIS in education as a powerful measurement tool, because it has the functionality of many discrete systems built into one package that can be viewed within LabView.

To enhance the understanding of the topics investigated in the lab, it would have been useful to control the servo motor using the myRIO as a discrete embedded system, with no communication from LabView once programmed. This is how an embedded would operate in some real-world scenarios after all, such as safety critical systems, where that embedded system is relied upon to operate a single task.

**References**

[1] M. Rouse, "Embedded System" in *Internet of Things Agenda,* December 2016. [Online]. Available: http://internetofthingsagenda.techtarget.com/definition/embedded-system. Accessed on: Nov. 7th, 2017.

[2] Unknown Author, "Embedded Systems – Overview" in *Tutorials Point*, Unknown Date. [Online]. Available: https://wnw.tutorialspoint.com/embedded_systems/es_overview.html. Accessed on: Nov. 7th, 2017.

[3] Unknown Author, "What is an FPGA?", in *Embedded Micro,* Unknown date. [Online]. Available: https://embeddedmicro.com/tutorials/mojo-fpga-beginners-guide/what-is-an-fpga. Accessed on: Nov. 7th, 2017.

[4] Unknown Author, "What is Data Acquisition?" in *National Instruments,* Unknown Date. [Online]. Available: http://www.ni.com/data-acquisition/what-is/. Accessed on: Nov. 9th, 2017.