Lab 4: Introduction to LabVIEW

EG-UY 1003 Y1B

Jason Yao

Team members: Alice (Nanda) Ross, Jeremiah Nofrada

Date of experiment: 16th July 2015

Date due: 19th July 2015

**Abstract**

The objective of this lab was to design three digital systems by utilizing the LabVIEW programming software. These three systems included a simple calculator, a lighting control system for a building, and a thermal control system. The three digital systems were successfully built, and tested on a circuit board to ensure it ran correctly. This lab was significant because it demonstrated the capability of utilizing simulated instruments in a laboratory, instead of physical instruments. The lab also introduced basic programming concepts that would be used in future labs.

**Introduction**

In this lab, three digital systems were to be graphically modeled using the LabVIEW programming software. These systems would then be tested for functionality and usability in hardware tests.

An important part of this lab was the LabVIEW software, defined by the EG Lab Manual (2015) as “a development environment for creating graphical programs… that simulate actual laboratory instruments”. LabVIEW thus enables engineers to develop and iterate quickly in software, instead of relying on physical prototyping that may be much more resource intensive, and may be cost prohibitive. As such, LabVIEW is an important tool to initially design and test systems, and allows the system to be changed drastically without adverse effects (loss of physical equipment, cost overruns, etc.).

The LabVIEW program does this digital designing by creating **virtual instruments (VIs).** Each VI is composed of two parts, a front panel, and a back panel. The **front panel** can be thought of as the user’s interface, allowing a user to interact with the VI by reading any output, and permitting the user to change the inputs. The **back panel** can be thought of as a mathematical function – something that takes in a number of inputs, and gives an output after transforming the inputs in some fashion.

Both panels are built from component parts – for front panels, they are constructed from **controls** (inputs) and **indicators** (outputs from the VI). For back panels, they are represented as block diagrams, containing **structures**, **functions**, and **terminals**. These are all examples of **nodes**, which are connected to one another via wires. Structures and functions are the objects used to transform any inputs from the front panels, which then outputs to the terminals, which are linked to the indicators in the front panel.

In this lab, the **National Instruments’ Educational Laboratory Virtual Instrumentation Suite** (NI-ELVIS) board was used to interface between the software and the physical world. The NI-ELVIS is an advanced circuit board that allowed users to quickly see whether their program worked as well on hardware as it did in software. In this lab, the NI-ELVIS was used to demonstrate that the thermal control and lighting control systems would work in a hardware setting.

The objectives of the lab were to design and build three digital logic systems, including a simple calculator, a thermal control system, and a lighting control system. The objectives of the experiment were achieved, and all three systems were functional.

**Procedures**

Due to the high amount of procedural elements in this lab, only the procedures used to build the thermal control system would be included. To build this system, the front panel needed to be initialized first, since it was a common factor regardless of whether the system was in automatic or manual mode. The front panel included a thermometer in control mode, three switches (labeled AC, Heater, and Manual/Auto respectively), along with three indicator LEDs.

With the front panel completed, the back panel needed to be designed for the two use cases. The first use case involved the system set to manual mode – in this mode, if the ac or heater buttons were turned on, the respective systems turned on as well, regardless of the current temperature. The second use case involved the system set to automatic mode – in this case, regardless of whether the heater or AC switches were on, the underlying AC or heating systems only turned on outside of the temperature range of 60°F to 80°F. With these two use cases, in the back panel, there needed to be a Boolean case statement as the centre of the back panel logic. Surrounding this case statement were the six terminals from the controls and indicators in the front panel (three from switches, three from indicator LEDs), along with the fan control module, heat control module, and temperature control modules required to interface with the NI-ELVIS board.

In manual mode, the indicator LED for whether the system was in manual mode had to be turned on. After that, the heater and AC systems do the same thing, linking to their respective indicator LED terminals before finally linking to their respective hardware module.

In automatic mode, the indicator LED for whether the system was in manual mode had to be turned off. The temperature was then fed through some digital logic blocks in order to ascertain if it was outside of the pre-set boundaries. If the temperature was greater than 80°F, then the AC indicator LED would turn on, and the AC module would start up. If the temperature was lower than 60°F, then the heater LED would turn on, and the heater module would start up. In this fashion, a simple but robust thermal control system was implemented in both hardware and software.

After the system was assembled, it was tested on the NI-ELVIS board, to ensure that the digital logic was sound when implemented with hardware. The system tested whether indicator LEDs turned on correctly, and whether the underlying AC and heating modules were set correctly in each of the different modes.

**Data/Observations**

For the simple calculator, as shown in Fig. 1, the user interface (UI) was much simpler than other calculators, though it could still do the basic functionalities of adding, subtracting and multiplying. The system was thus easy to use, with no frills to distract away from functionality. One thing that should be noted was the way floating-point values (numbers with decimals) were calculated, because the answer was truncated after 5 decimal places. This meant that when 0.200003 was added to 1.57, the resulting answer was 1.77.

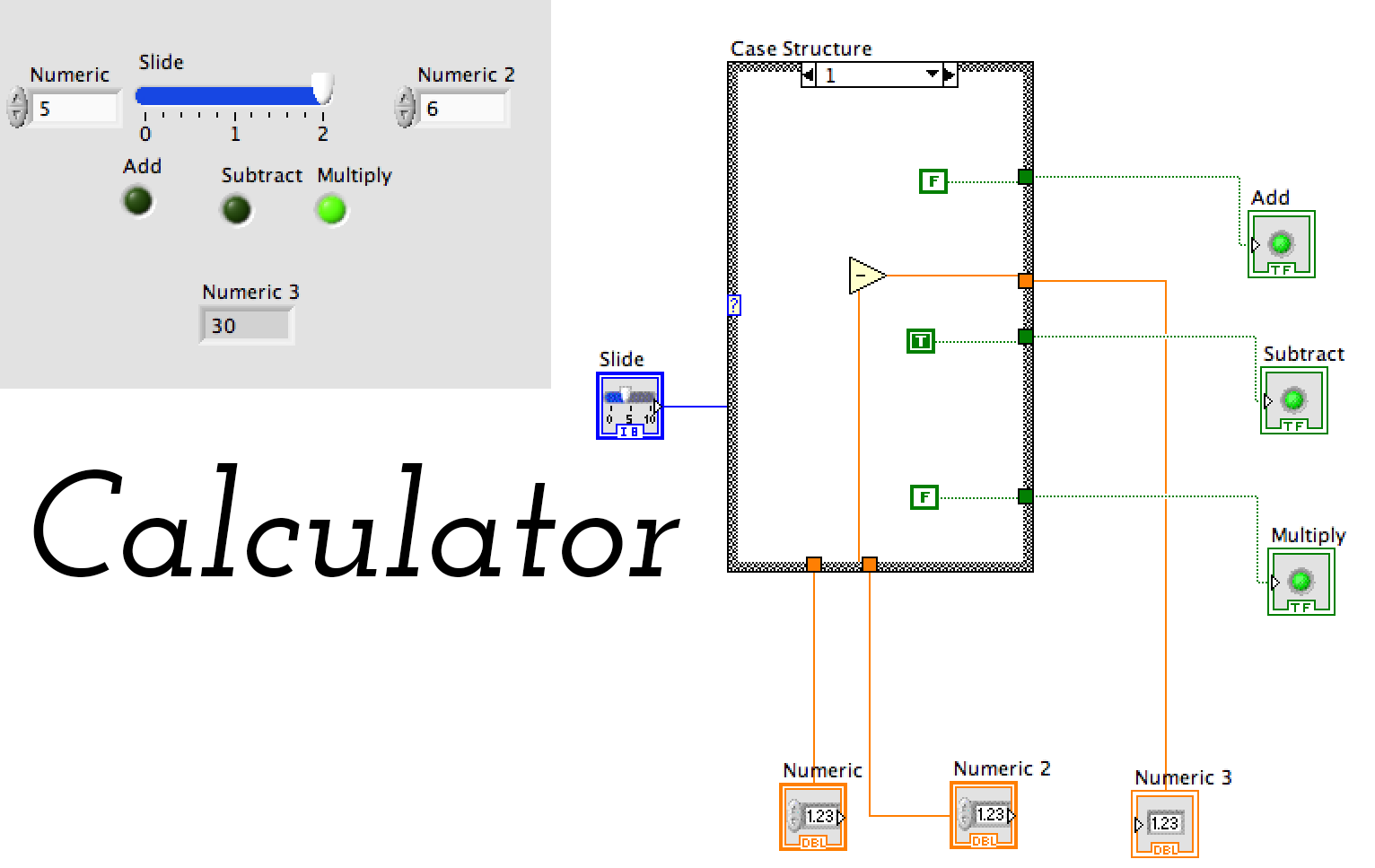


Fig. 1: Simple calculator front & back panels

For the thermal control system, the UI was again very plain, as shown in Fig. 2, but was responsive to the changes in input when in automatic mode, as changes in the input temperature resulted in immediate responses from the heating/cooling subsystems. The system was thus easy to use, with no frills to distract away from functionality.

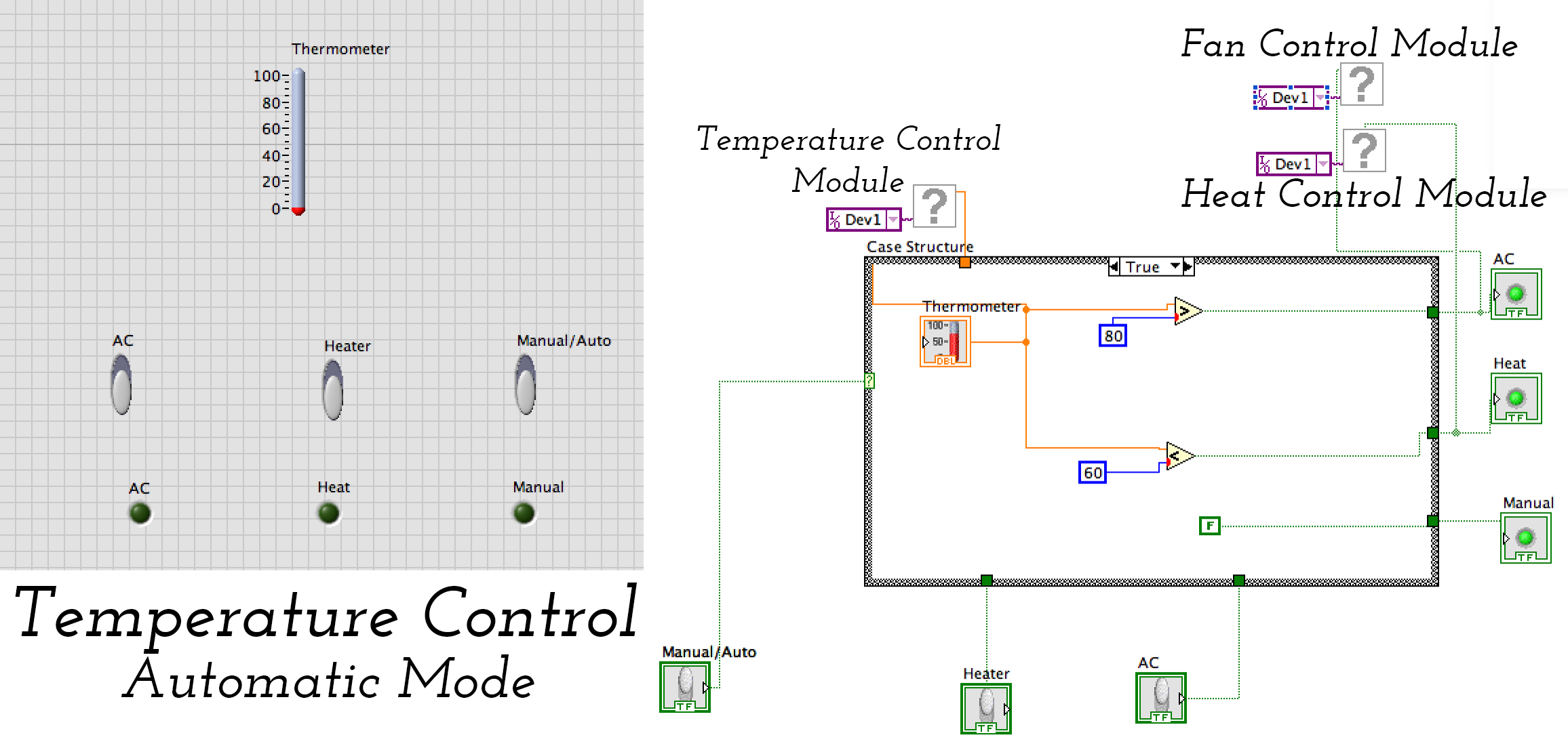


Fig. 2: Thermal control front & back panels

For the lighting control system, the UI was again plain, as shown in Fig. 3, but was functional. System responsiveness was quick, and followed the required specifications. The system was thus easy to use, with no frills to distract away from functionality.

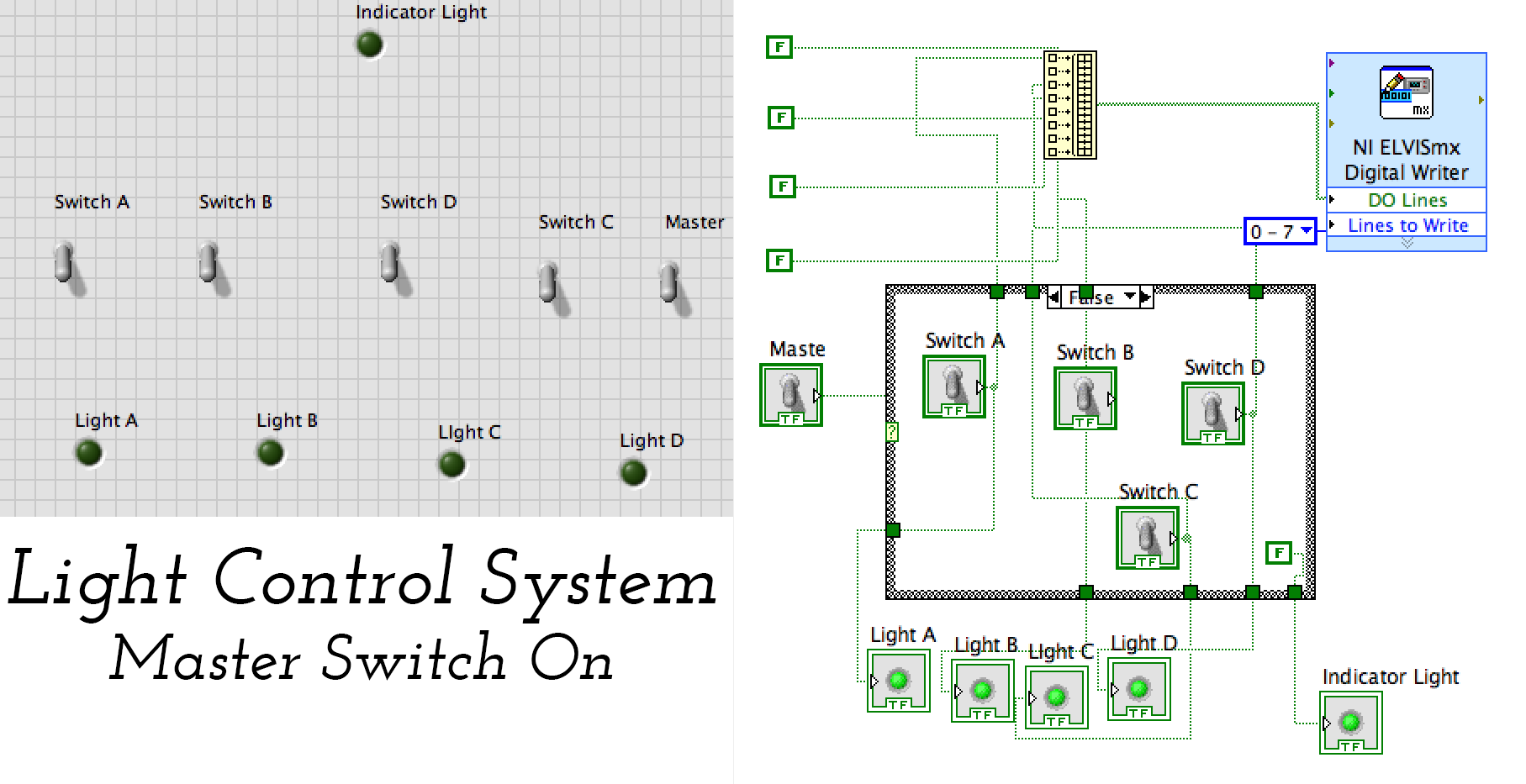


Fig. 3: Lighting control front & back panels

**Discussion/Conclusions**

One of the largest advantages in using LabVIEW included having a “birds eye view” of how the overarching digital system looked like, which gave engineers an easy understanding of how all the system components were supposed to work together.

Another advantage in using LabVIEW was the ability to simulate many lab instruments in software, instead of having to physically connect these instruments together. For example, instead of having 30 specific physical instruments, only the computer would be required to model these instruments, allowing for much faster iteration when prototyping.

However, there were some disadvantages when LabVIEW was used - the fact that LabVIEW abstracted away basic systems-level programming concepts made it so that if a LabVIEW constructed program failed, there was limited data available explaining why it had failed. This was in direct contrast with text-based programming languages, in which the programmer/engineer had fine-tuned granular control over which parts of the system to abstract away.

A specific example of this was during the calculator system design, in which floating-point values were handled differently than nominal hardware standards. In text-based programming languages, floating-point numbers follow the IEEE 754 standard hardware specification. Under this standard, floating-point values were given a precision of 20 bits. This meant that whenever the LabVIEW calculator computed any calculations, it would give a different answer than other machines would, due to handling these floating-point calculations differently. In real-world terms, if banking systems utilized the LabVIEW calculator for their computations, fractions of pennies and dimes would be lost with every transaction. This may not sound like much, but when millions of transactions occur daily, these small fractions add up over time, eventually resulting in system failure, external exploitation, or millions of dollars simply lost from the system.

In this lab, all three systems functioned correctly, responded quickly, and were intuitive to work with. Thus, the three system designs were successful in both functionality and in user experience.

A way to improve the lab in future teachings could be the addition of button inputs to the calculator design, allowing digits, negative and decimal symbols to be represented in hardware. This would allow for a more familiar calculator design, enabling a smoother user experience.

**Works Cited**

NYU Polytechnic School of Engineering. 2015. “Lab 4: Introduction to LabVIEW”. EG 1003 Online Lab Manual. Accessed 18 July 2015 from manual.eg.poly.edu.

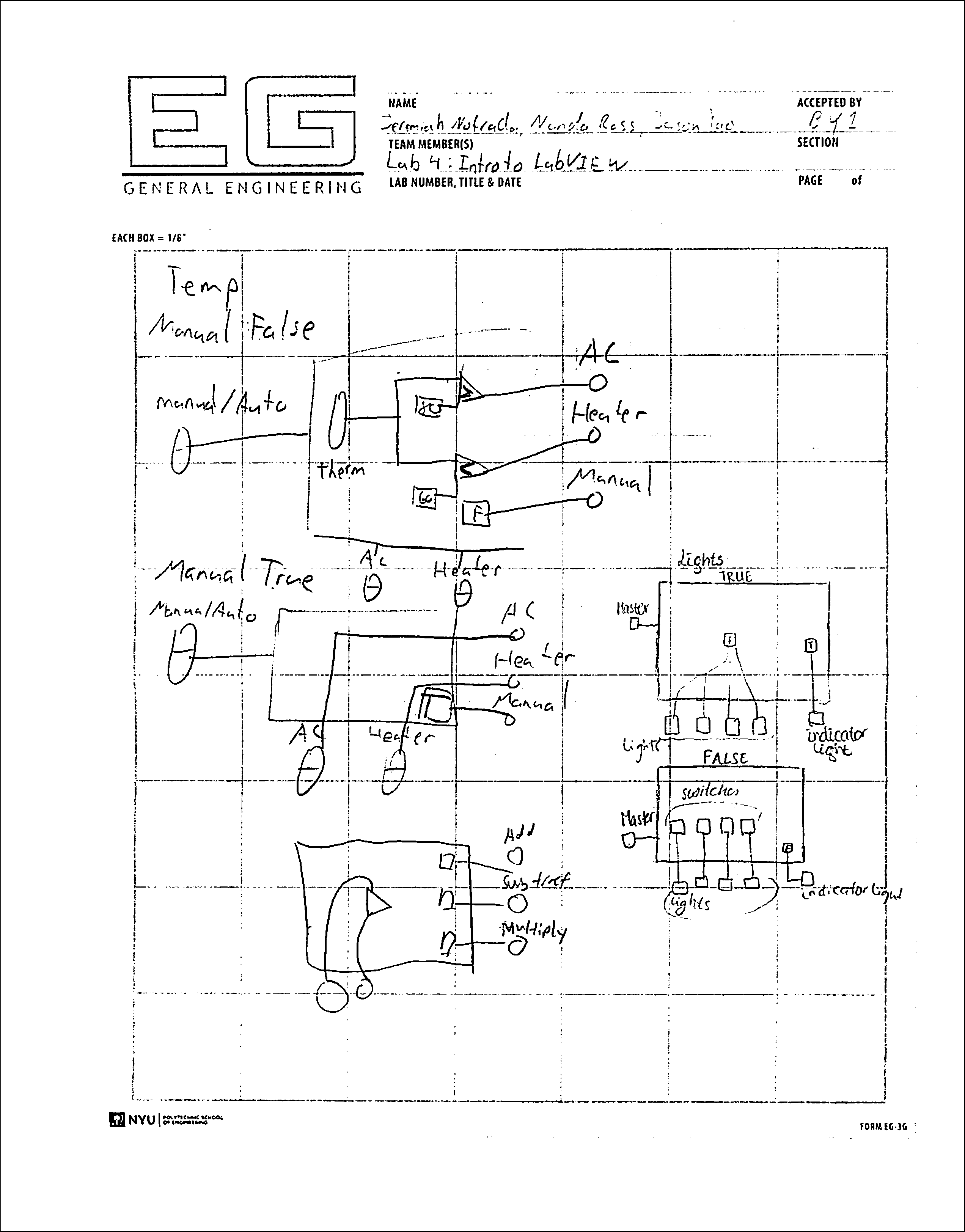
**

Fig. : Initial design sketch