LabVIEW and Arduino as a gateway to PLC programming

Dr. Wesley B. Williams P.E., University of North Carolina, Charlotte

Dr. Wesley B. Williams is an assistant professor in the Department of Engineering Technology and Construction Management at the University of North Carolina at Charlotte. Dr. Williams joined the department in 2011, teaching courses in the mechanical engineering technology and electrical engineering technology undergraduate programs as well the facilities management graduate program. His research interests include additive manufacturing, instrumentation, controls, and lapping. His professional experience also includes work in the pharmaceutical industry where he designed custom instrumentation and automation solutions for pharmaceutical researchers. In addition to his professional and academic activities, Dr. Williams is active mentoring students through programs such as the UNC Charlotte Senior Design Program and US FIRST Robotics.

LabVIEW and Arduino as a gateway to PLC programming

Abstract

The key topics of an instrumentation and controls course are a) the transducers that convert the phenomenon of interest into a predictable signal b) the actuators that influence the process in question and c) the hardware and software that control these outputs based the measured inputs. Common candidates for these control systems are data acquisition (DAQ) systems, microcontrollers, and programmable logic controllers (PLCs). Outside of the electrical engineering technology curriculum, engineering technology undergraduates at the University of North Carolina at Charlotte do not cover any of these devices until taking a single course junior or senior year. In a graduate course focused on facilities instrumentation and controls, National Instruments DAQ systems, Arduino microcontrollers, i-TRiLOGI ladder logic simulator and Allen Bradley industrial PLCs were introduced in that order to quickly bring students up to speed on the strengths, weaknesses, and suitable applications for each device.

The graduate students in the class came from both civil engineering and construction management undergraduate programs. As such, they had limited backgrounds in electronics and programming. This assessment was reflected in an initial course survey where students rated their abilities in electronics theory, electronics hands on, LabVIEW, and MATLAB as being weak, slightly weak, or average (the three lowest categories on the survey). Conversely, they rated their abilities in math and physics as being average, strong, or very strong (the three highest categories on the survey).

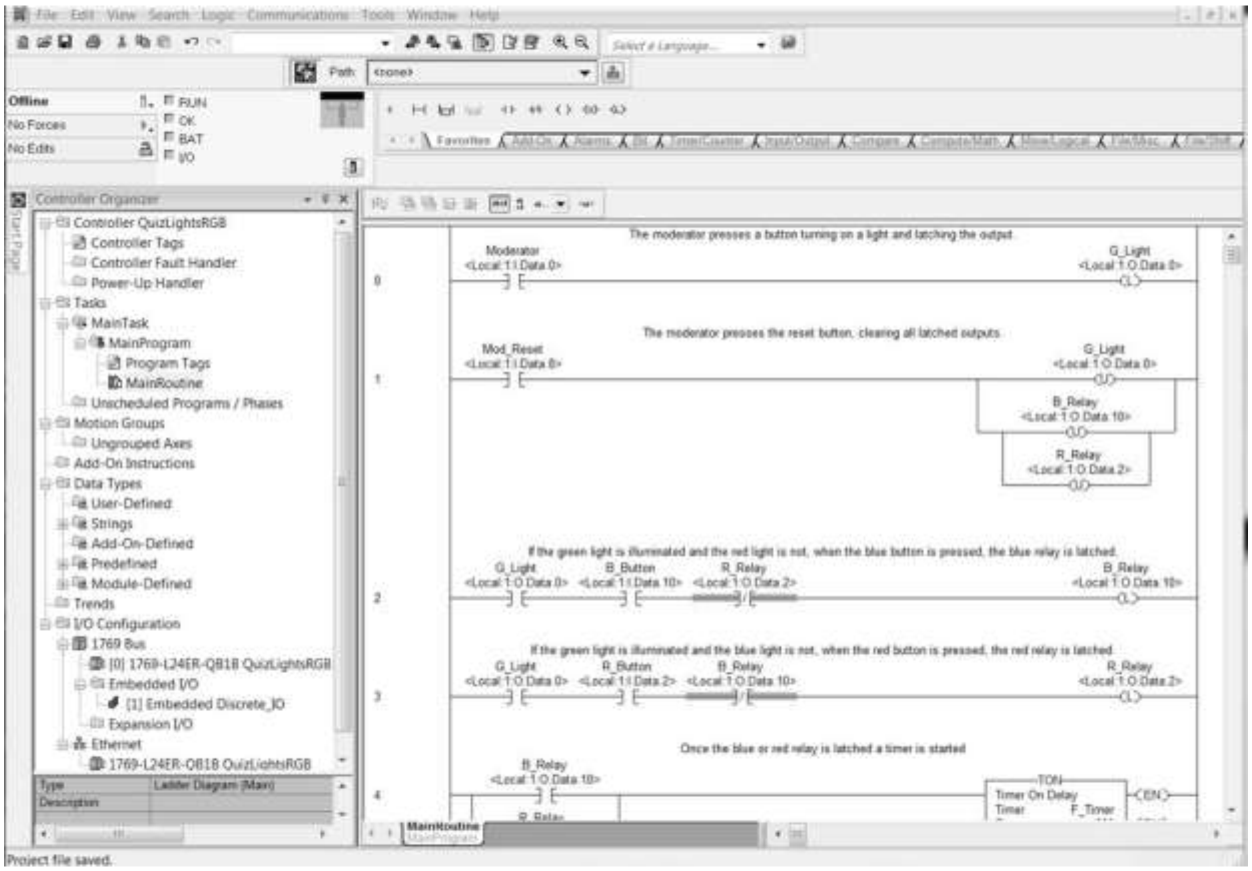
National Instruments LabVIEW and a NI USB-6009 multifunction DAQ were chosen as a starting platform given the graphical programming environment, readily available tutorials, and integration with the hardware. Students were shown demonstrations involving simple breadboard circuits before the students worked in small groups to replicate and extend the LabVIEW code and breadboard wiring to include both measurements and control circuits. After completing the breadboard exercises with LabVIEW and the USB DAQ, students were introduced to the Arduino 1.0 Integrated Development Environment (IDE) and an Arduino Uno microcontroller. As a high level programming language, students viewed the Arduino sketches as easy to follow given a limited introduction in lecture and access to the Arduino website. The abundance of published sketches made it easy for students to see the versatility of the microcontroller platform and learn common coding structures. Other key skills that were acquired through the Arduino exercises included effective documentation of code, addressing inputs and outputs, and understanding the difference between analog outputs (available on the NI USB DAQ) and the emulated analog outputs utilizing PWM (available on the Arduino Uno).

Ladder logic was introduced with i-TRiLOGI ladder logic simulators. Relay ladder logic was presented with a focus on the similarities to the graphical, 'wiring' approach to programming found in LabVIEW. The Arduino skills of addressing inputs and outputs and documenting code

were also reemphasized as was a focus on program/logic flow. Allen Bradley PLCs programmed with RSLogix 5000 software were the final platform introduced to the students, applying their acquired knowledge of ladder logic from i-TRiLOGI to industry grade hardware and software. Throughout Arduino, i-TRiLOGI, and RS Logix 5000, a common exercise was used to highlight the similarities, differences, and capabilities of each platform.

Introduction

Programmable logic controllers are a common solution to automation and control projects for engineers. Despite their frequent application, PLC instruction is limited to a single upper division course for Mechanical Engineering Technology students at the University of North Carolina at Charlotte. The versatility and capability of many industrial PLCs though comes at the expense of accessibility, with a programming environment that can be intimidating for students with limited programming experience (Figure 1) or students accustomed to languages like C++ and Python[1]. To overcome this intimidation with students from mechanical engineering or facilities management backgrounds, students were led through a common task using a selection of software and hardware that was believed to be more accessible, to build their skills and confidence prior to programming with RSLogix 5000.



```
0  The moderator presses a button turning on a light and latching the output
   Moderator <Local:1:I.Data.0>                                G_Light <Local:1:O.Data.0> (L)
1  The moderator presses the reset button, clearing all latched outputs
   Mod_Reset <Local:1:I.Data.8>                                G_Light <Local:1:O.Data.0> (U)
                                                               B_Relay <Local:1:O.Data.10> (U)
                                                               R_Relay <Local:1:O.Data.2> (U)
2  If the green light is illuminated and the red light is not, when the blue button is pressed, the blue relay is latched.
   G_Light <Local:1:O.Data.0>  B_Button <Local:1:I.Data.10>  R_Relay <Local:1:O.Data.2>   B_Relay <Local:1:O.Data.10> (L)
3  If the green light is illuminated and the blue light is not, when the red button is pressed, the red relay is latched
   G_Light <Local:1:O.Data.0>  R_Button <Local:1:I.Data.2>  B_Relay <Local:1:O.Data.10>   R_Relay <Local:1:O.Data.2> (L)
4  Once the blue or red relay is latched a timer is started
   B_Relay <Local:1:O.Data.10>                                 TON Timer On Delay  Timer F_Timer  (EN)
   R_Relay
```

Figure 1. Screen shot of the RSLogix 5000 software, which while powerful and versatile can appear quite daunting to students with limited programming experience.

Students were initially introduced to the LabVIEW programming environment based on the graphical nature of the programming, the robust in program help, and the active community of users providing both tutorials and support[2]. Arduino was chosen as the second platform due to the readability of the code, the abundance of examples and tutorials[3], and the key lessons of program flow that can be taught with loops and conditionals. Given that Arduino was originally targeted a non-technical audience[4], the expectation was that this would be an accessible environment for the students to develop programming skills. The final step before moving to RSLogix 5000, was introducing the students to ladder logic with i-TRiLOGI, which provides a much simpler environment given the more limited toolset.

In order to give these new programmers structure in their exercises, tasks in each programming environment were broken down to the following steps:

- Define Inputs/Outputs: Determine names/aliases and physical connection assignments
- Generate Code Elements: Produce the commands for discrete tasks in the program
- Test Code Elements: Test the discrete tasks and revise as necessary
- Assemble Code Elements: Integrate the working elements of code into a cohesive program

The exercise

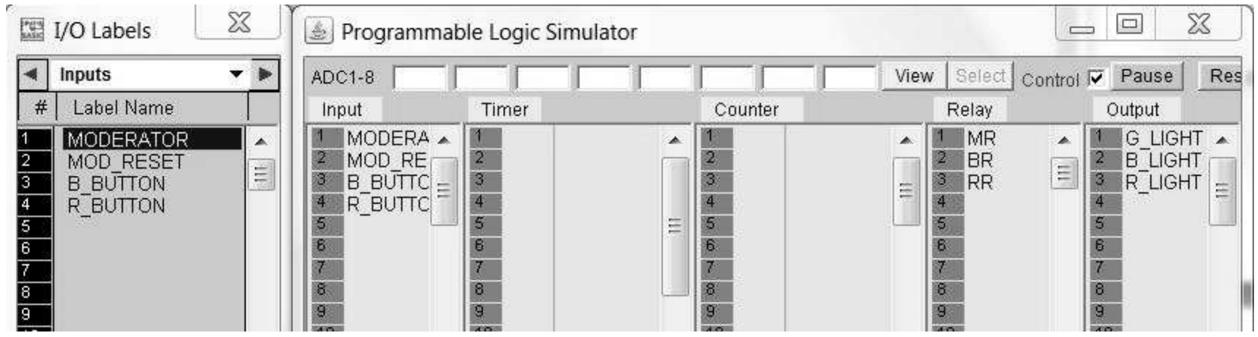
Students were guided through developing the code to simulate a quiz show with two buzzers. The moderator would press a button as she started to read the question, turning on a green light, which would stay on. The first player (blue or red) to press their button would have their light come on and flash. The second player to press their button would not see their light illuminate or flash. Once a particular question was finished, the moderator would press a reset button, turning off all lights in preparation for the next question.

The definition of inputs and outputs takes place in different ways depending on the programming environment. In Arduino, text commands take care of this by assigning aliases to the various numbered pins and setting those pins up as either inputs or outputs (Figure 2). The Arduino code also includes a number of variables that are used in the code (trgState and pbState for example, which are used to denote the whether the moderators switch has been flipped or the blue players button has been pressed).

```
// declaration of variables
const int GLED=8; // green LED - the trigger light
const int BLED=9; // blue LED  light for the blue player
const int RLED=10; // red LED  light for the red player

const int MODER=7; // moderator switch
const int BS=6; // blue player switch
const int RS=5; // red player switch

int trgState=0;  // initial trigger state
int pbState=0;  // blue player win state
int prState=0;  // red player win state
int d=10; //sets the delay in milliseconds


void setup() {
  // sets the pins to inputs and outputs
  pinMode(GLED, OUTPUT); pinMode(BLED, OUTPUT); pinMode(RLED, OUTPUT);
 // sets up the LED pins as outputs

  pinMode(MODER, INPUT); pinMode(BS, INPUT); pinMode(RS, INPUT);
  //sets up the switch pins as inputs

  digitalWrite(GLED,LOW);  // sets all the LEDs to LOW
  digitalWrite(BLED,LOW); digitalWrite(RLED,LOW);
}
```

Figure 2. Definition of inputs and outputs in Arduino.

Definition of inputs and outputs in i-TRiLOGI takes place through filling out an input/output (I/O) table. The point was emphasized with students that just as a programmer cannot use a variable that has not been defined previously in Arduino, tags for switches, outputs, and relays are not available in i-TRiLOGI's ladder logic unless they appear in the I/O table. There are four buttons used in the i-TRiLOGI code, two for the moderator and one for each player. Each player and the moderator are also assigned a relay and light in the I/O table.

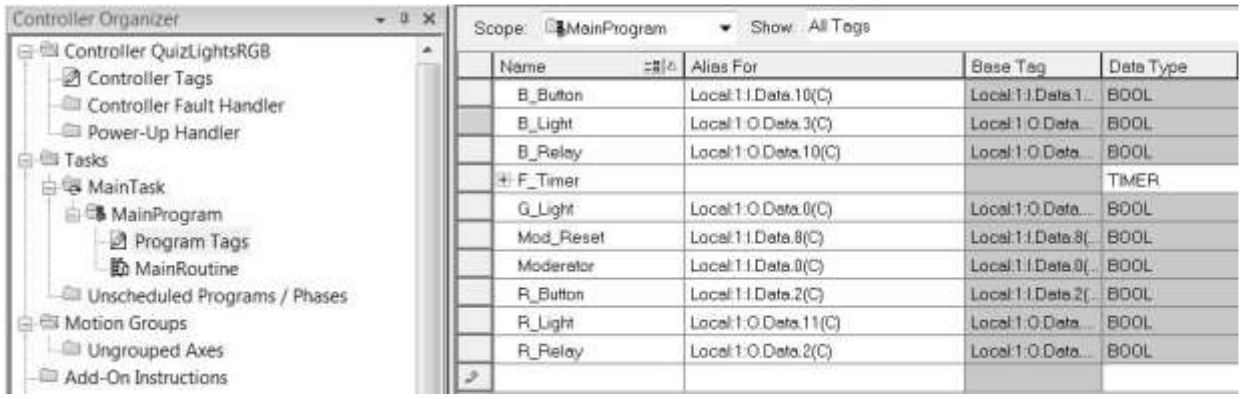


#	Label Name
1	MODERATOR
2	MOD_RESET
3	B_BUTTON
4	R_BUTTON
5	
6	
7	
8	
9	

#	Input	Timer	Counter	Relay	Output
1	MODERA			MR	G_LIGHT
2	MOD_RE			BR	B_LIGHT
3	B_BUTTC			RR	R_LIGHT
4	R_BUTTC				
5					
6					
7					
8					
9					

Figure 3. The I/O table used in i-TRiLOGI ladder logic software.

The I/O definition for RSLogix 5000 is similar to i-TRiLOGI with the programmer filling out a table under program tags to establish the inputs, outputs, and any timers or counters that are used. In this program there is a button, relay, and light for each player (e.g., B_Button, B_Relay, B_Light). The moderator has a button, a reset button, and a green light. There is also a timer (F_Timer) that is used to make the lights flash (Figure 4).



Scope: MainProgram Show: All Tags

Name	Alias For	Base Tag	Data Type
B_Button	Local:1:I.Data.10(C)	Local:1:I.Data.1...	BOOL
B_Light	Local:1:O.Data.3(C)	Local:1:O.Data...	BOOL
B_Relay	Local:1:O.Data.10(C)	Local:1:O.Data...	BOOL
F_Timer			TIMER
G_Light	Local:1:O.Data.0(C)	Local:1:O.Data...	BOOL
Mod_Reset	Local:1:I.Data.8(C)	Local:1:I.Data.8(...	BOOL
Moderator	Local:1:I.Data.0(C)	Local:1:I.Data.0(...	BOOL
R_Button	Local:1:I.Data.2(C)	Local:1:I.Data.2(...	BOOL
R_Light	Local:1:O.Data.11(C)	Local:1:O.Data...	BOOL
R_Relay	Local:1:O.Data.2(C)	Local:1:O.Data...	BOOL

Figure 4. Program tags table in RSLogix 5000.

With the inputs and outputs successfully mapped, the next task in each program was to generate elements of the code. These elements of code were small portions of the overall code, dealing with specific inputs and outputs that could be isolated and tested individually. As an example, the code to handle the green, moderator light is shown below in Arduino (Figure 5), i-TRiLOGI (Figure 6), and RSLogix (Figure 7). The Arduino code uses a digitalRead command to determine if the moderator switch has been flipped. If it has, the variable trgState is set to 1, for use in later logic in the program and the green LED is turned on with a digitalWrite command.

```
void loop() {

  if (digitalRead(MODER)==HIGH) {  // triggers the green LED once the switch is flipped
    trgState=1;digitalWrite(GLED,HIGH);
  }
  else {
   trgState=0;
  }
```

Figure 5. Arduino code for determining when to illuminate the green light.

The first rung of the ladder logic in i-TRiLOGI accomplishes the task by using a normally open moderator switch, a normally open reset switch, a moderator relay (MR), and the green light. When the MODERATOR button is pressed, the MR relay is energized, latching the rung, which will keep the relay energized and the green light lit until the MOD_RESET button is pressed. It is worth noting that i-TRiLOGI has the capacity for latching outputs, but this example was specifically done with the relay in parallel to show how to construct a latching rung (Figure 6).

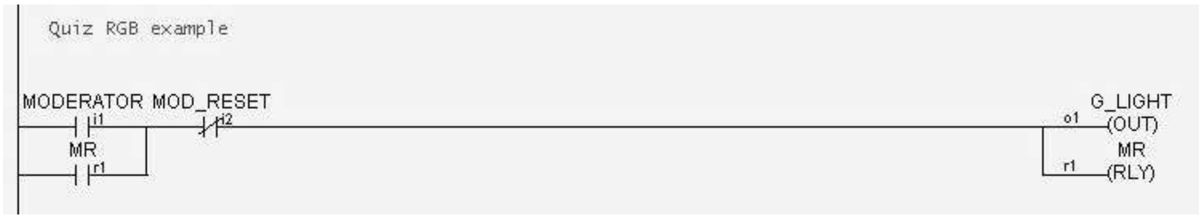


Figure 6. First rung of the i-TRiLOGI program latching the green light once the moderator button is pressed.

The ladder logic is similar in RSLogix, but uses two rungs and takes advantage of latching outputs. Rung 0 simply energizes and latches the green light (G_Light) output when the moderator button is pressed. Rung 1 unlatches the G_Light as well as relays to indicate whether the blue or red player pressed their button first (B_Relay and R_Relay) when the Mod_Reset button is pressed (Figure 7). The operation of the blue and red lights was separated out into relays and lights, because unlike the green light, the blue and red lights needed to flash which would be cumbersome if latching the lights was also being used within the logic of the code (to exclude the other players button press).This unlatching of all latched outputs returns the program to the initial ready state in preparation for another question.

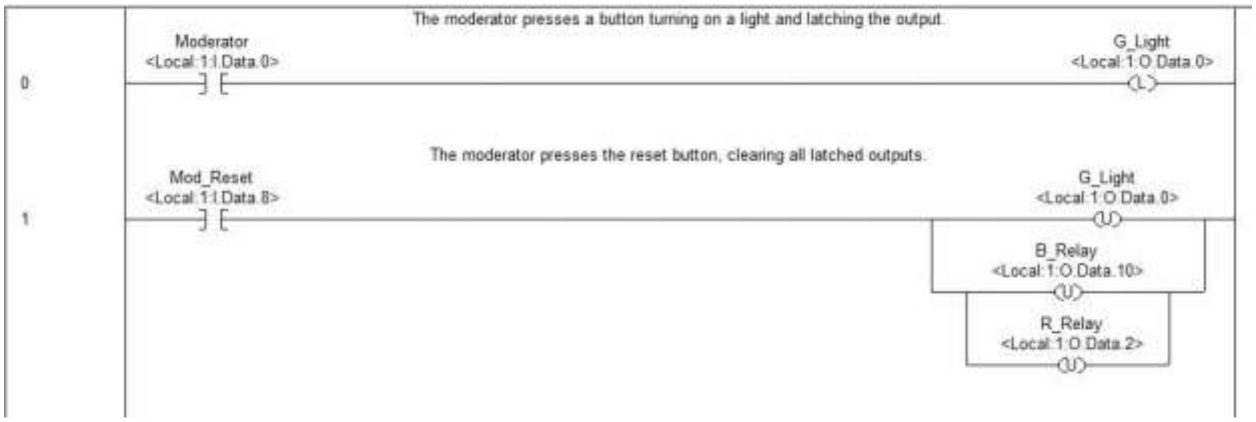


Figure 7. RSLogix, Rung 0 and 1, showing the green light latching and unlatching operation.

Individual elements of code, like those shown above, were then tested to ensure proper operation. The Arduino code was tested with a simple breadboard circuit using three LEDs, a switch for the moderator and push buttons for the blue and red players. The i-TRiLOGI code was simulated to verify the correct operation of the rungs as they were built. The RSLogix ladder logic was verified on CompactLogix Package Controller with 16 digital inputs and 16 digital outputs, where button inputs were simulated by forcing them with the software.

As these snippets of code were generated and tested, they were assembled into the final code that is found in the Appendices. By using the same program through each programming language, students were able to see how each language handled certain elements of the program. For example, in the case of the flashing lights the Arduino code handled that with a simple loop that turned the light on then off after a delay. The i-TRiLOGI code used the internal clock functions which provide intermittent connectivity between the energized relays and the appropriate lights. In RSLogix, the same effect was achieved by using a repeating timer[5] and a less than conditional statement prior to the energized lights.

Assessment

Students found the introduction to data acquisition and control through LabVIEW to be accessible. Given online video tutorials[2], a lab session, and the robust help functions in the program, they were able to replicate and extend the in class demonstrations with limited intervention on my part. Most questions on those exercises revolved around building the associated circuits on the breadboard, though sampling frequency DAQ assistant also came up on occasion.

The students' response' to Arduino were more mixed. While students did well with the exercise presented in this paper and seemed to genuinely enjoy the other Arduino projects in the class, they worked at a much slower pace than they did with LabVIEW. Tasks that involved more advanced programming concepts like nested loops, switch cases, and arrays were challenging for them. In these cases the students showed an ability to explain and evaluate code that used these concepts, but they were slow to independently implement these techniques into their code when it would have been the appropriate/efficient approach.

The introduction of ladder logic through i-TRiLOGI was well received by the students. By outlining their code on paper and filling in their I/O tables before starting to program their first rungs, students had some time to think about and reflect on their program strategy. This made the actual programming go much quicker, with students quickly arriving at ladder logic that was close to the behavior that they wanted. Errors that appeared in simulations were quickly investigated and alternatives were generated and tested. Much of this success continued into RSLogix as well, but the additional menu choices and depth of customization provided more opportunities for students to get lost or an errant click to give unintended behavior.

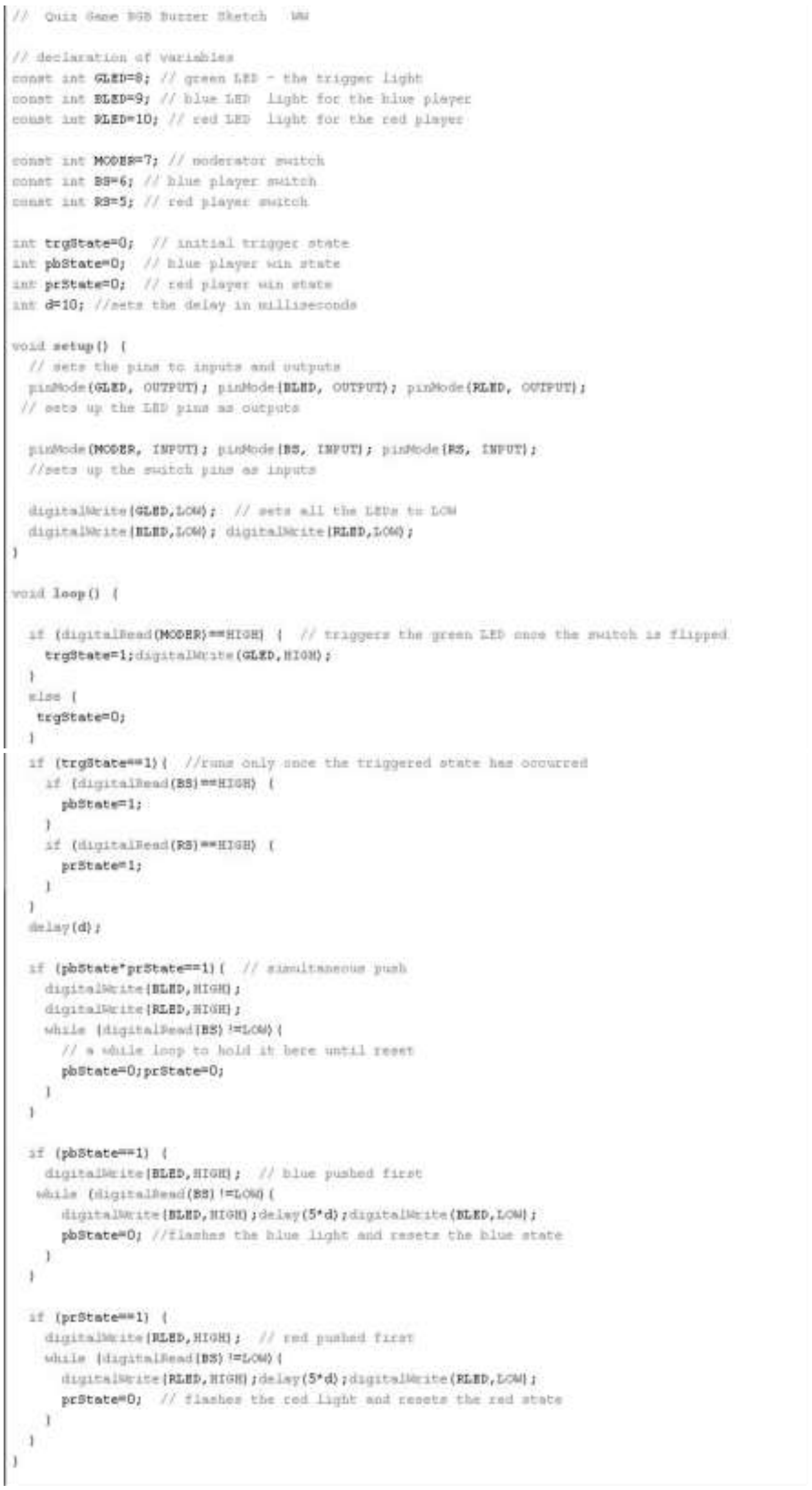
In surveys at the end of the class, students reported that they felt more comfortable with ladder logic than LabVIEW or Arduino. They cited the ability to focus in on the single rung affecting a misbehaving output as a tremendous asset when troubleshooting a program. Students also felt as though they would likely come across ladder logic and PLCs in the careers after graduation. LabVIEW, while also a graphical program, was largely viewed as more difficult to follow with the program flow between blocks being more haphazard, especially with poor layouts and commenting. In discussions about Arduino, students noted that while they enjoyed the activities greatly and it made programming and electronics more accessible, they did not see themselves using it professionally on a project. Despite this, over half the group did feel that Arduino would be something that they would revisit on their own as a low cost way to experiment with electronics, programming, and automation. This type of response gives encouragement to educators who are increasingly incorporating Arduino into their electronics lab activities, both on campus and remotely instructed[6].

It is unclear if the Arduino and LabVIEW exercises at the beginning of the class were instrumental in the ready acceptance of PLC ladder logic. It is possible that the visual structure of ladder logic makes it inherently more accessible than text based or block based programming. The students in the class were also not 'heavily burdened' with preexisting notions about programming, given their limited initial programming experience. The reinforcement of good programming practices such as defining inputs and outputs, generating and testing small segments of code, and liberally commenting the code regardless of programming environment, did seem to yield better code as the semester progressed. Finally, stepping through a common exercise for each programming environment was useful as a vehicle to compare and contrast both the capabilities and syntax of each programming language. This left the students understanding that each combination of hardware and software brought varied capabilities to the project, but they were ultimately just tools to implement the logical control that they devised.

Bibliography

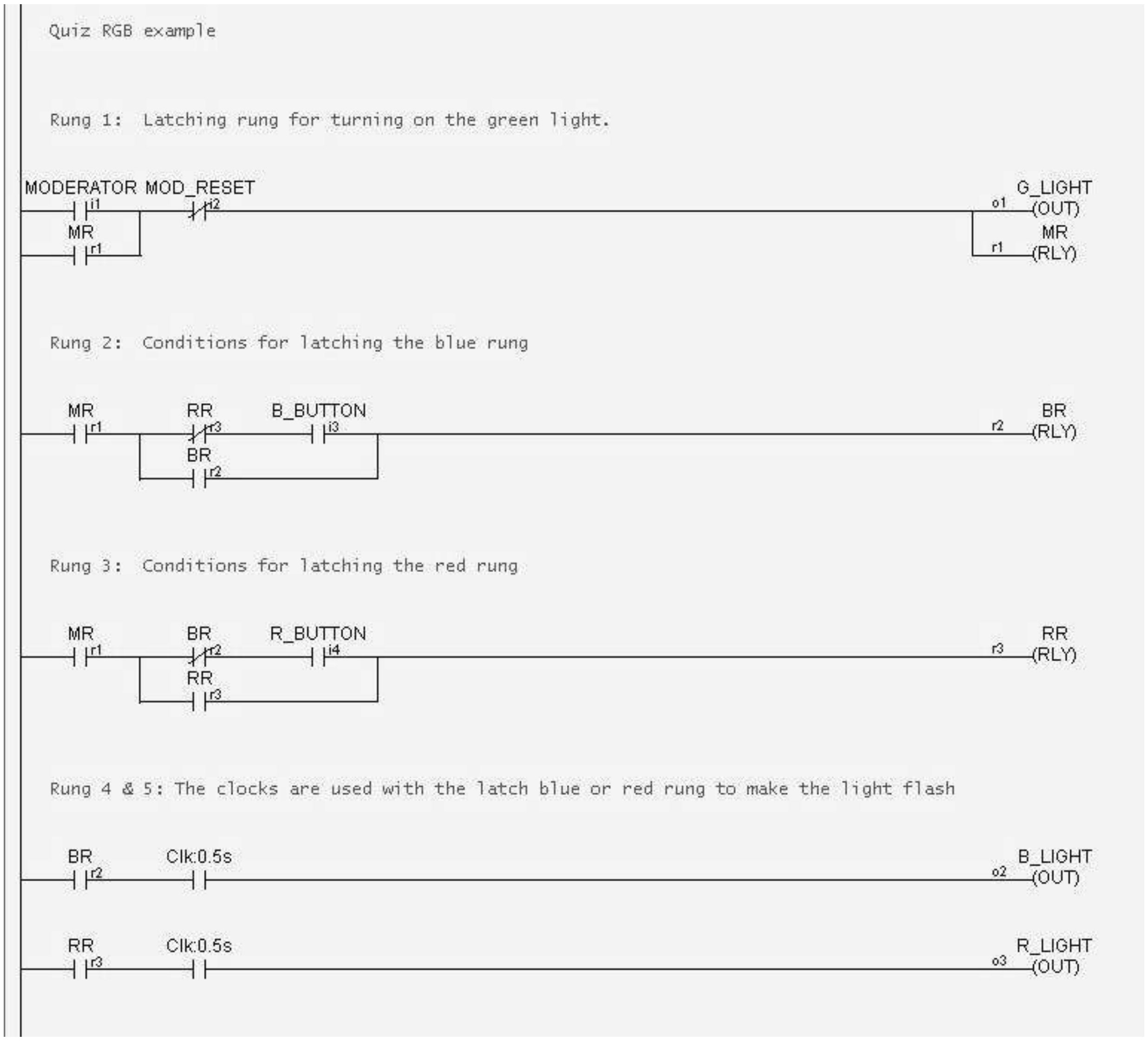
1. FOSTER, M. HAMMERQUIST, C. A Review of Programmable Logic Controllers in Control Systems Education. *American Society for Engineering Education Annual Conference*. (2010)
2. NI LabVIEW 101: Video Instruction for Students. http://www.ni.com/academic/students/learnlabview/
3. www.arduino.cc/en/Tutorial/HomePage
4. BANZI, M. *Getting Started with Arduino, 2nd ed*. O'Reilly Media, Sebastopol, CA, 2011.
5. REHG, J. SARTORI, G. Instructional Algorithms Enhance Student Understanding of PLC Ladder Logic Programming. *American Society for Engineering Education Annual Conference*. (2010)
6. SARIK, J. KYMISSIS, I. Lab Kits Using the Arduino Prototyping Platform, *40th ASEE/IEEE Frontiers in Education Conference*. (2010)

Appendix A: Arduino Code



```
//  Quiz Game RGB Buzzer Sketch   WW

// declaration of variables
const int GLED=8; // green LED - the trigger light
const int BLED=9; // blue LED  light for the blue player
const int RLED=10; // red LED  light for the red player

const int MODER=7; // moderator switch
const int BS=6; // blue player switch
const int RS=5; // red player switch

int trgState=0;  // initial trigger state
int pbState=0;  // blue player win state
int prState=0;  // red player win state
int d=10; //sets the delay in milliseconds

void setup() {
  // sets the pins to inputs and outputs
  pinMode(GLED, OUTPUT); pinMode(BLED, OUTPUT); pinMode(RLED, OUTPUT);
 // sets up the LED pins as outputs

  pinMode(MODER, INPUT); pinMode(BS, INPUT); pinMode(RS, INPUT);
  //sets up the switch pins as inputs

  digitalWrite(GLED,LOW);  // sets all the LEDs to LOW
  digitalWrite(BLED,LOW); digitalWrite(RLED,LOW);
}

void loop() {

  if (digitalRead(MODER)==HIGH) {  // triggers the green LED once the switch is flipped
    trgState=1;digitalWrite(GLED,HIGH);
  }
  else {
   trgState=0;
  }
  if (trgState==1){  //runs only once the triggered state has occurred
    if (digitalRead(BS)==HIGH) {
      pbState=1;
    }
    if (digitalRead(RS)==HIGH) {
      prState=1;
    }
  }
  delay(d);

  if (pbState*prState==1){  // simultaneous push
    digitalWrite(BLED,HIGH);
    digitalWrite(RLED,HIGH);
    while (digitalRead(BS)!=LOW){
      // a while loop to hold it here until reset
      pbState=0;prState=0;
    }
  }

  if (pbState==1) {
    digitalWrite(BLED,HIGH);  // blue pushed first
   while (digitalRead(BS)!=LOW){
      digitalWrite(BLED,HIGH);delay(5*d);digitalWrite(BLED,LOW);
      pbState=0; //flashes the blue light and resets the blue state
    }
  }

  if (prState==1) {
    digitalWrite(RLED,HIGH);  // red pushed first
    while (digitalRead(BS)!=LOW){
      digitalWrite(RLED,HIGH);delay(5*d);digitalWrite(RLED,LOW);
      prState=0;  // flashes the red light and resets the red state
    }
  }
}
```

Appendix B: i-TRiLOGI code


Quiz RGB example

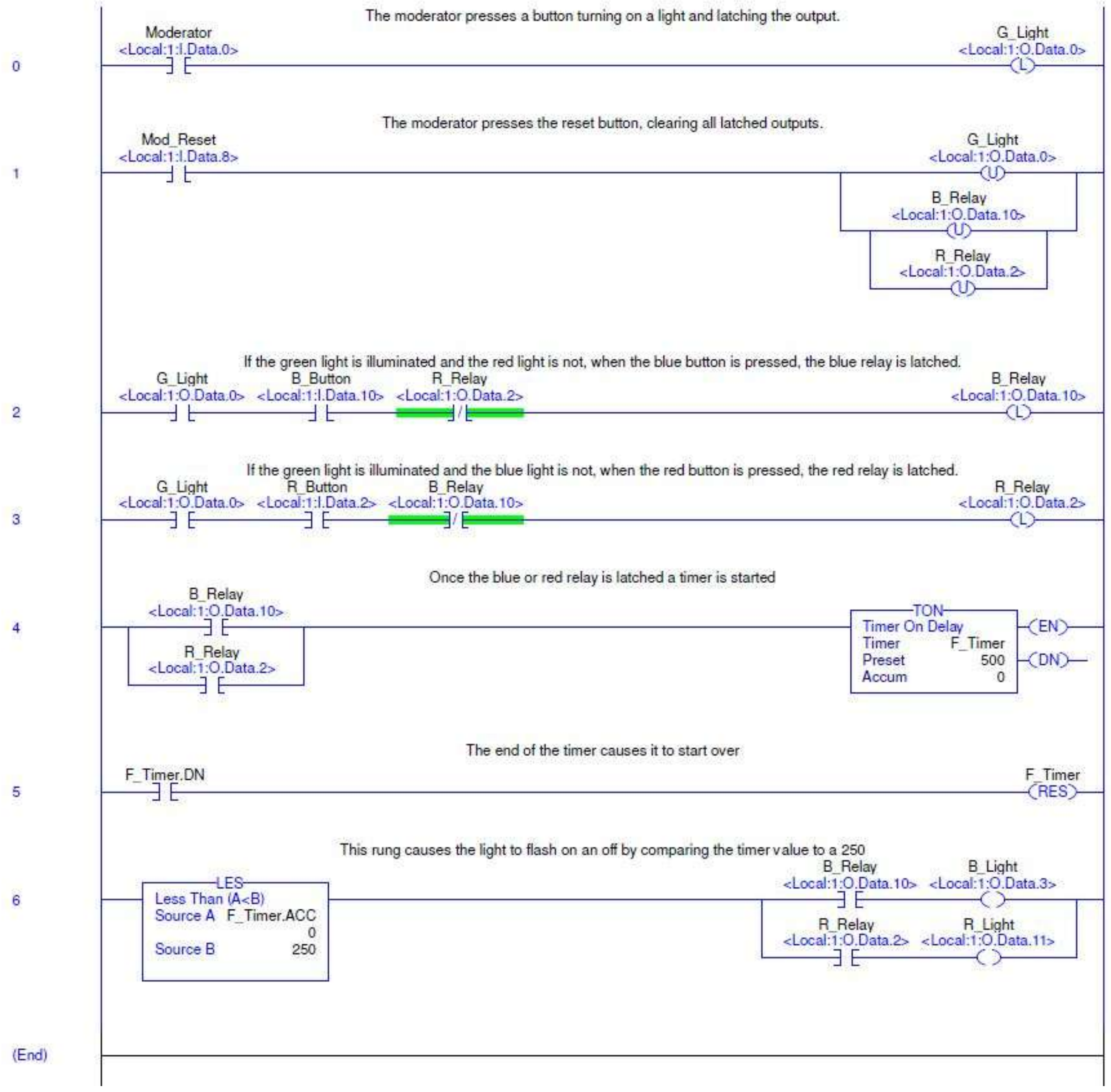
Rung 1: Latching rung for turning on the green light.

Rung 2: Conditions for latching the blue rung

Rung 3: Conditions for latching the red rung

Rung 4 & 5: The clocks are used with the latch blue or red rung to make the light flash

Appendix C: RSLogix Code


The moderator presses a button turning on a light and latching the output.

The moderator presses the reset button, clearing all latched outputs.

If the green light is illuminated and the red light is not, when the blue button is pressed, the blue relay is latched.

If the green light is illuminated and the blue light is not, when the red button is pressed, the red relay is latched.

Once the blue or red relay is latched a timer is started

The end of the timer causes it to start over

This rung causes the light to flash on an off by comparing the timer value to a 250