Laboratory Environment Monitoring, Safety and Control System with GSM Interface

Presented by Brian L. Decker

EGT 417 SENIOR DESIGN IN TECHNOLOGY

NORTHERN KENTUCKY UNIVERSITY

SENIOR PROJECT INSTRUCTOR: DR. Mark Kesh

ACADEMIC ADVISOR: Dr. Gang Sun

DECEMBER 01, 2018

Abstract

The proposed project consists of the design, code development and hardware implementation of a microcontroller-based laboratory environment monitoring and robot safety cage control system. The embedded system uses an 8-bit AVR RISC-based microcontroller to process input from two independent subsystems: an environmental sensor suite to include light level, temperature and humidity, and infrared presence sensing devices to determine if individuals are within the footprint of the robot safety cage. In response to the input collected from the subsystems via a TTL logic bus, the controller will display parameters to a local OLED display, arm/disarm the power supply of the robot system in the safety cage and send alerts via GSM text messages to designated cellular phones (lab manager, NKU maintenance, NKU security) when preconfigured parameters are exceeded. Additionally, a text-parsing algorithm will allow remote arm/disarm commands, system status updates and inquires of current environmental data to be sent via GSM from authorized cellular phones without the need for a separate application. The system will also be constructed using a modular hardware and software schema allowing for the provided functionality to be easily reconfigured to support other scenarios of use. An example would be the replacement of the presence sensors with input from an anemometer and rain sensor allowing the system to serve as a remote weather station that would send text alerts if a wind gust exceeded a set value or an hourly transmission of current conditions. By replacing the 120-volt AC power supply with a solar array and rechargeable battery, the system could be placed in a remote area and, due to the use of global GSM, maintain all functionality without the need for any external connections.

Acknowledgements

I would like to acknowledge the faculty and staff of NKU for their continuous support and encouragement. Specific thanks to Dr. Morteza Sadat-Hossieny for his encouragement to apply to CINSAM, resulting in a grant for my initial research, and to Dr. Gang Sun whose knowledge and experience were invaluable. I would also like to thank Pam Kremer and Roger Miller for their willingness to coordinate meeting times, schedule appointments and grant access to the manufacturing and computer labs.

I would also like to make a personal acknowledgement to my wife, Hadley, whose support (both personally and financially) has allowed me to participate in this program and achieve my life-long dream of becoming an engineer.

Corporate and Academic Sponsors

The NKU Center for Integrative natural Science and Mathematics supported this project as part of the summer 2018 undergraduate research initiatives.

Seeed Studio provided support for the project through their prototype PCB program which offers significant discounts for student projects and associated hardware that meets size and complexity requirements. The L.C.M.S. qualified for the program by utilizing double sided PCB's with poured ground planes and overall sizes less than 100mmx100mm.

Digi-Key Corporation provided support for the project by waiving shipping and handling fees following a request for participation in Academic Support program.

Table of Contents

Table of Figures

Introduction

The L.M.C.S. is intended to implement several important functions in one unit that allows for both local and regional conductivity via a GSM data connection. While it is true that the functions of safety control and environmental status seem to be unrelated – the desire that both functions include the ability for remote access, control and reporting creates a point of commonality that makes their combination sensible. By utilizing the power of inexpensive and easily available modern microcontrollers which may easily be programmed in the "C" language, it is possible to integrate the functions of multiple sensors – those utilized for safety as well as environmental sensing – into one comprehensive control and communication system capable of processing both local and remote I/O. OSHA has clearly defined the need for a safety cage environment to protect those working near a robotic system. Studies in Sweden and Japan indicate that many robot accidents do not occur under normal operating conditions but, instead during programming, program touch-up or refinement, maintenance, repair, testing, setup, or adjustment. During many of these operations the operator, programmer, or corrective maintenance worker may temporarily be within the robot's working envelope where unintended operations could result in injuries [3]. The use of presence sensing devices to disable power to the robot when anyone enters the safety-cage while the L.M.C.S. is operating in the run state will aid in the prevention of these types of accidents. In addition to the immediate benefit of accident prevention, the GSM text notifications will alert appropriate NKU faculty and staff to occurances of potentially unsafe operating conditions in the lab. This will allow for a more meaningful interaction among students, lab manager and engineering faculty with quantifiable and objective data regarding these situations as opposed to subjective reporting or rumors of "near-misses".

Definition of the problem

The expectations of the Laboratory Environment Monitoring, Safety and Control System (L.M.C.S.) for the NKU robotic systems lab are threefold: OSHA requires a safeguard system to prevent movement of robot arms and/or manipulators while a person is present within the operating envelope of a robotic system or safety cage [1]. Because the environmental conditions of the lab - temperature, humidity and light levels - are often important factors of processes conducted within the lab (the application and curing of specialized coatings, adhesives or protective films for example) such conditions should be monitored [2]. Finally, it is important for the lab manager, NKU security and others as determined by faculty to have remote access to reports of the current lab environment parameters and immediate notification of breaches of the safety cage by unauthorized persons and/or occurrences of environment parameters that exceed pre-set values.

Proposed Solution

My proposal is to perform the design, code development and hardware implementation of a microcontroller-based laboratory environment monitoring and robot safety cage control system with a GSM interface. This system (L.M.C.S.) will use an 8-bit AVR RISC-based microcontroller to process input from two independent subsystems: environmental data to include light level, temperature and humidity, and infrared presence sensing devices to determine if individuals are within the robot safety cage. In response to the input collected from the subsystems, the L.M.C.S. will display parameters to a local OLED display, illuminate or extinguish LED status indicators, arm/disarm the power supply of the robot system in the safety cage and send alerts via GSM text

messages to designated cellular phones (lab manager, NKU maintenance, NKU security) when preconfigured parameters are exceeded. Additionally, a text-parsing routine will allow remote arm/disarm commands, L.M.C.S. status updates and inquires of current environmental data to be sent via GSM from authorized cellular phones without the need for a separate application. To perform the presence sensing function of the system, two light curtains will be constructed – each utilizing four IR retro-reflective sensors. To integrate the light curtains into the overall L.M.C.S. schema, a custom PCB controller will be designed and built to provide power to the sensors and transmit trigger signals to the main L.M.C.S. controller. While each of the individual functions presented above have been implemented in a variety of standards throughout industry, during my document review while preparing this proposal I was unable to find another occurrence of these functions combined along with the inclusion of the text-parsing system to allow for remote inquiry and system control in a similar integrated unit as in the L.M.C.S.

Prior Work on the Project

The general principles of operation, sensors, components and control schema proposed for the L.M.C.S. have been standardized and are common throughout industry. To provide for the highest level of reliability in a system intended to provide for the safety of NKU students and faculty in the robot lab operating environment, it is critical to use only components, sensors and code standards that have been tested and engineered to meet industry and OSHA standards. There are several prior systems that share common aspects of the control system proposed for the L.M.C.S. that were produced to at least the prototype form – however, none of them share the combination of robotic safety control, environmental monitoring and GSM communications.

The systems that most nearly align in terms of general operation are commercial security systems that utilize a cellular data interface for transmitting alarms or security status information to owners, police and/or fire departments.

Many of the systems use a communication method other than GSM – one example is the basis for research produced by Hasan, Khan et al that uses a Bluetooth device to control the system via smartphone. A similar system which provides for alarm functions without the environmental sensing but utilizing GSM communication was presented in a paper by Olarewaju, Ayodele et. Al – "In this project we design and construct an automatic home security system based on GSM technology and embedded microcontroller unit. The system consisted of an infrared motion detector and a magnetic sensor as transducers, and the signals are then processed by an embedded microcontroller unit which activate the GSM module and send SMS messages to a mobile phone device [3]."

In contrast to that system, which is primarily intended as only an intrusion alarm, research focused instead on a GSM interfaced environmental monitroing system intended to observe the environmental data in a grain storage system. "A remote temperature and humidity monitoring system is designed based on GSM technology and digital sensors DSB1820 and SHT11 to monitor the temperature and humidity of the granary. These parameters can be adjusted with the controlling system to adapt various working conditions. Through the GSM system, the detected data could be sent to various monitoring devices, such as cellphones and laptops. These data can be used for data display, inquiry, controlling and storage at the remote terminals. The experimental results show that the system is convenient and concise, which meets the remote monitoring demand for the modern granary [4]."

Again – while there are examples of both research and commercial devices that incorporate portions of the functionality in the L.M.C.S., the combination of sensors, functionality and communication is believed to be unique.

Description of the Product/Process

The L.M.C.S. will consist of a standard locking non-metallic industrial enclosure containing the system 120VAC to 5VDC 2amp power supply, custom PCB, GSM module, external antenna, keyed system state switch, OLED display, audible tone generator, LED indicators and momentary pushbuttons to allow for user interface and inputs such as updates to the L.M.C.S. phone number data table, setting the system time or adjusting the display parameters for environmental data. Connections will be provided for 5V (Logic Level) data signals from the light, temperature, humidity and presence sensing devices and the control signal to the solid-state relay(s) handling control power to the robot.

In operation, the appropriate L.M.C.S. state – armed, disarmed, run, admin – will be selected by the setting of the multi-position keyed safety switch. The L.M.C.S. state will be displayed on the OLED screen and the appropriate status LED will illuminate. The following is a list of I/O actions for each L.M.C.S. state:

- **Armed:** When the key switch is in the "Armed" position or the "System Armed" text message has been sent from a designated cellular phone, the L.M.C.S. will enter the "Armed" state. The OLED will display a rotation of "System Armed", Temperature and Humidity, and Time and a RED LED will be illuminated. The SSR will be OFF. A signal from the presence sensors will generate a warning tone until the presence is removed, and a preconfigured GSM text

message will be transmitted to the designated cellular phones.

- **Disarmed:** When the key switch is in the "Disarmed" position or the "System Disarmed" text message has been sent from a designated cellular phone, the L.M.C.S. will enter the "Disarmed" state. The OLED will display a rotation of "System Disarmed", Temperature and Humidity, and Time and a GREEN LED will be illuminated. The SSR will be OFF. In this state GSM inquiries may be sent to the L.M.C.S., but it may not be remotely armed, and the presence sensors will not generate any tones or messages.
- **Run:** When the key switch is in the "Run" position the system will enter the "Run" state. The OLED will display "System Run" and a BLUE LED will be illuminated along with a double beep. The SSR will be ON and a signal sent from the presence sensors will switch the SSR off, disabling the robot and sound an alarm tone. In this state GSM inquiries may be sent to the L.M.C.S., but it may not be remotely armed, and the presence sensors will not generate any tones or messages.
- **Admin:** When the key switch is in the "Admin" position the system will enter the "Admin" state. The OLED will display the primary phone number programmed into the L.M.C.S. and a YELLOW LED will be illuminated along with a double beep tone. The SSR will be OFF and in this state GSM inquiries and commands are disabled. While in the "Admin" mode the four momentary pushbuttons on the control panel become active and allow the user to:
    - Set the time displayed on the OLED.
    - Change, add or delete phone numbers stored in the L.M.C.S. phone number data table ROM so that data will be preserved in the event of a power failure or system reset.
    - Change temperature display from Celsius to Fahrenheit.

Main PCB and 8-bit RISC Based Processor

The use of the Atmel ATmega 1284 microprocessor offers numerous advantages over other possible solutions. This processor is the most powerful available in a DIP format, which is an important consideration given the prototype nature of this device. By utilizing a socketed DIP package, I am able to burn and test multiple software solutions to different chips and simply swap them out of the main IC socket. By executing powerful instructions in a single clock cycle, the ATmega 1284 achieves throughputs approaching 1 MIPS per MHz, balancing power consumption and processing speed. In addition to processing speed, the ATmega 1284 combines 128KB ISP flash memory with read-while-write capabilities, 4KB EEPROM, 16KB SRAM, 32 general purpose I/O lines, 32 general purpose working registers, a real time counter, three flexible timer/counters with compare modes and PWM, two USARTs, a byte oriented 2-wire serial interface, an 8-channel 10-bit A/D converter with optional differential input stage with programmable gain, programmable watchdog timer with internal oscillator, SPI serial port, a JTAG (IEEE 1149.1 compliant) test interface for on-chip debugging and programming, and six software selectable power saving modes.

An additional strength of the ATmega 1284 is the RISC (reduced instruction set) based programming model that it utilizes. By using the Visual Micro plugin for Microsoft Visual Studio 2017, I am able to both write code in “C” and load that code to the microcontroller via the use of an FTDI equipped USB cable. The primary advantage of the RISC architecture is the efficiency gained by the use of local variables declared within a routine, as opposed to global variables that must be held in memory for extended periods of time. Because only a portion of RAM need be

allocated, optimization of the compiler provides for a lower power consumption during operation and a fast wake time when the controller is in a sleep mode.

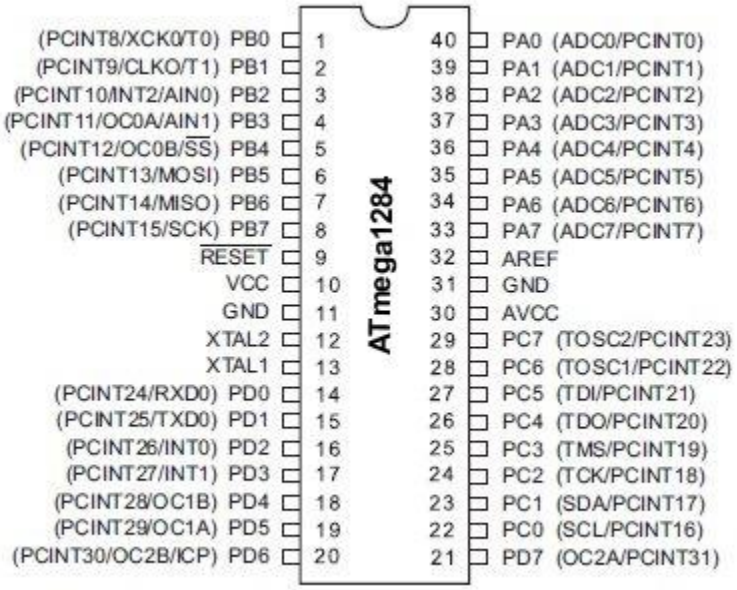


Figure 1 ATmega 1284 Pinout (Atmel Inc., 2018)

Schematic of the Product/Process

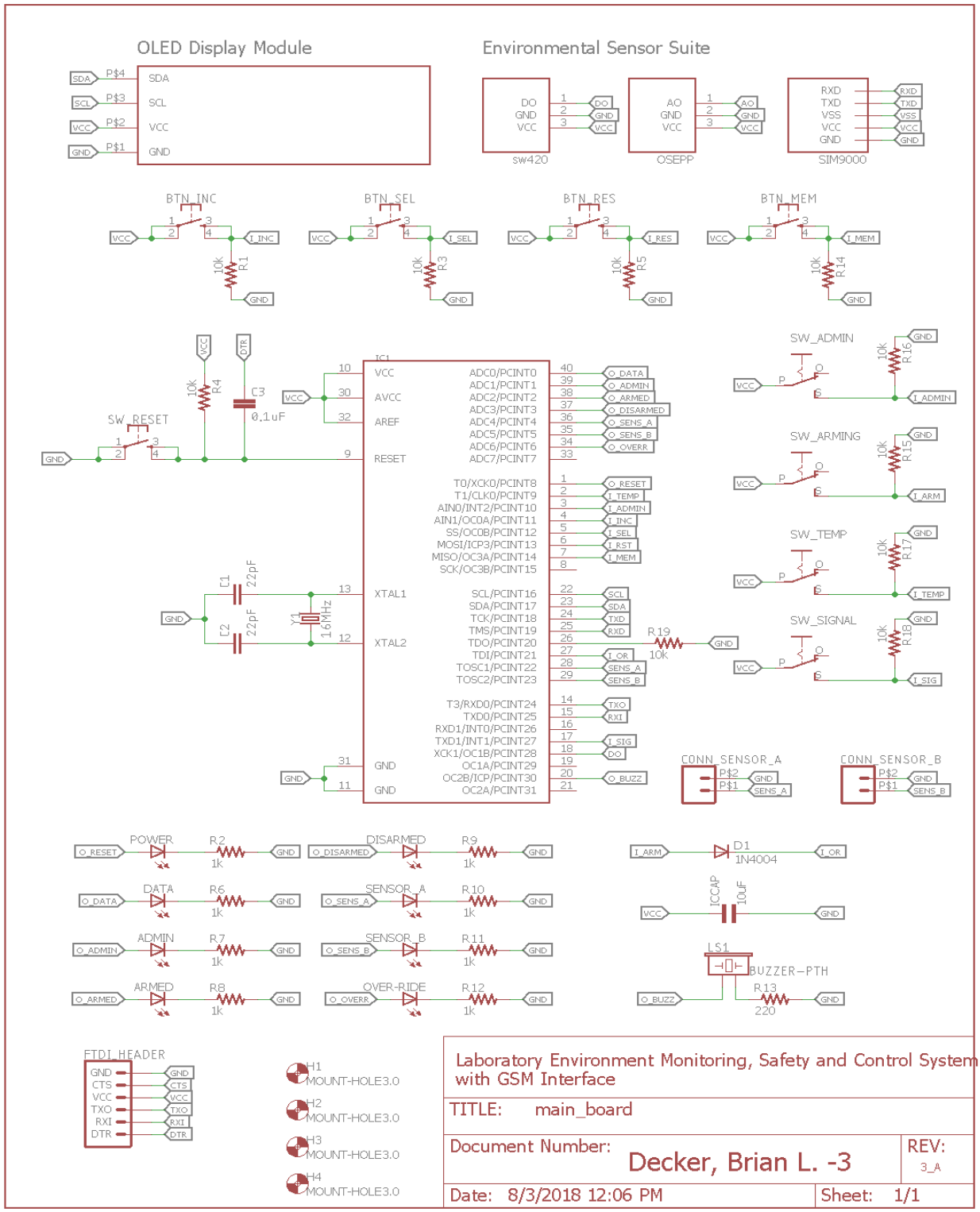


Figure 2 Main Control Schematic (Brian L. Decker, 2018)

Gerber file view of L.M.C.S. Main PCBs

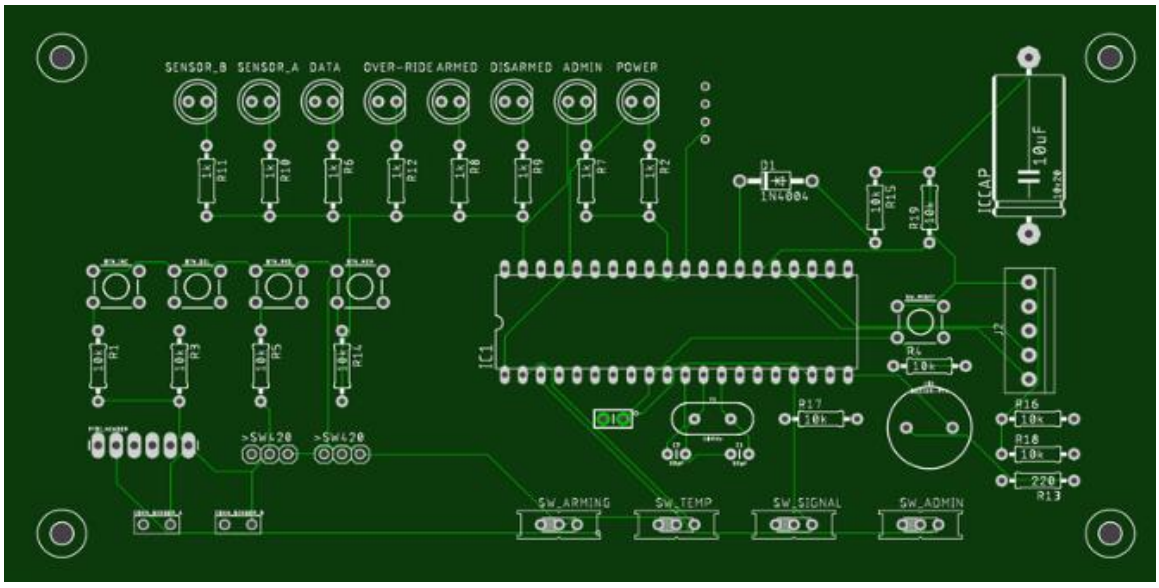


Figure 3 L.M.C.S. initial PCB Layout in SEEED Fusion Gerber Viewer (Brian L. Decker, 2018)

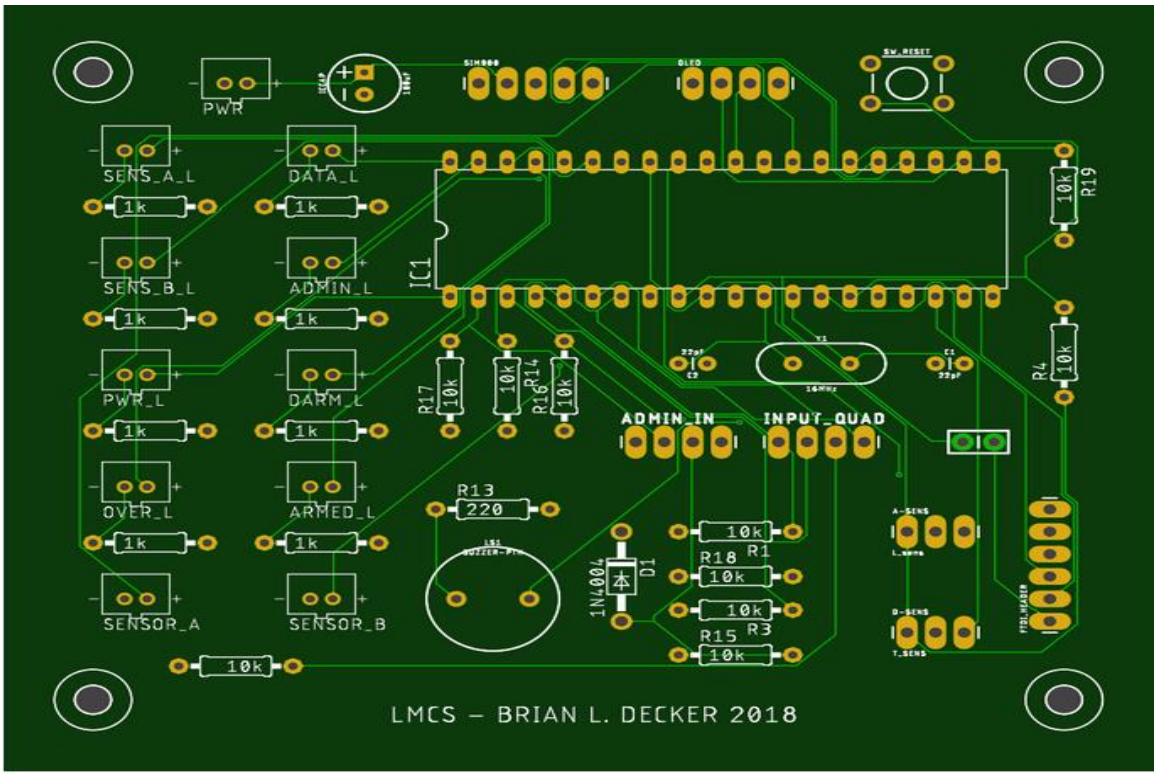


Figure 4 L.M.C.S. PCB Layout 2 in SEEED Gerber Viewer (Brian L. Decker, 2018)

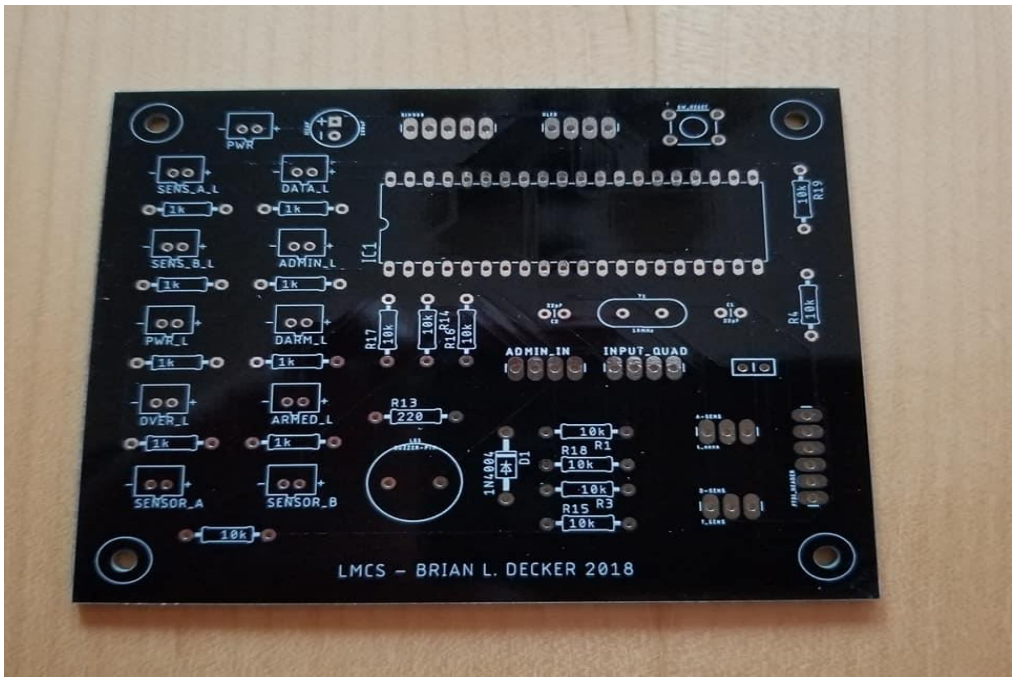


Figure 5 L.M.C.S. PCB initial production sample (Brian L. Decker, 2018)

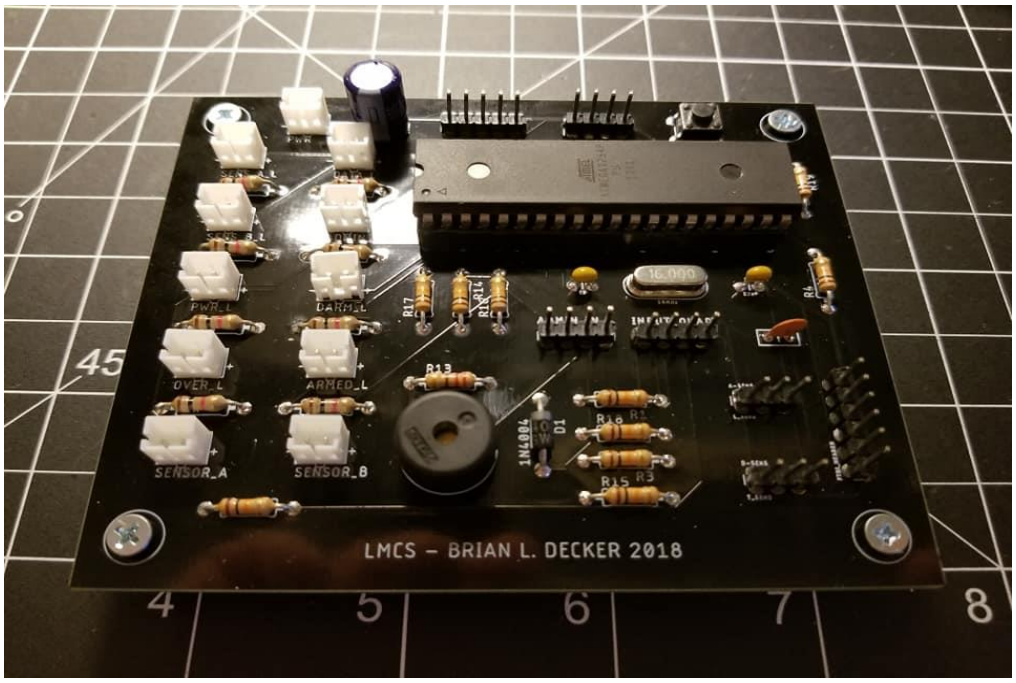


Figure 6 L.M.C.S. PCB assembled (Brian L. Decker, 2018)

Schematic of Light Curtain 4-1 sensor control

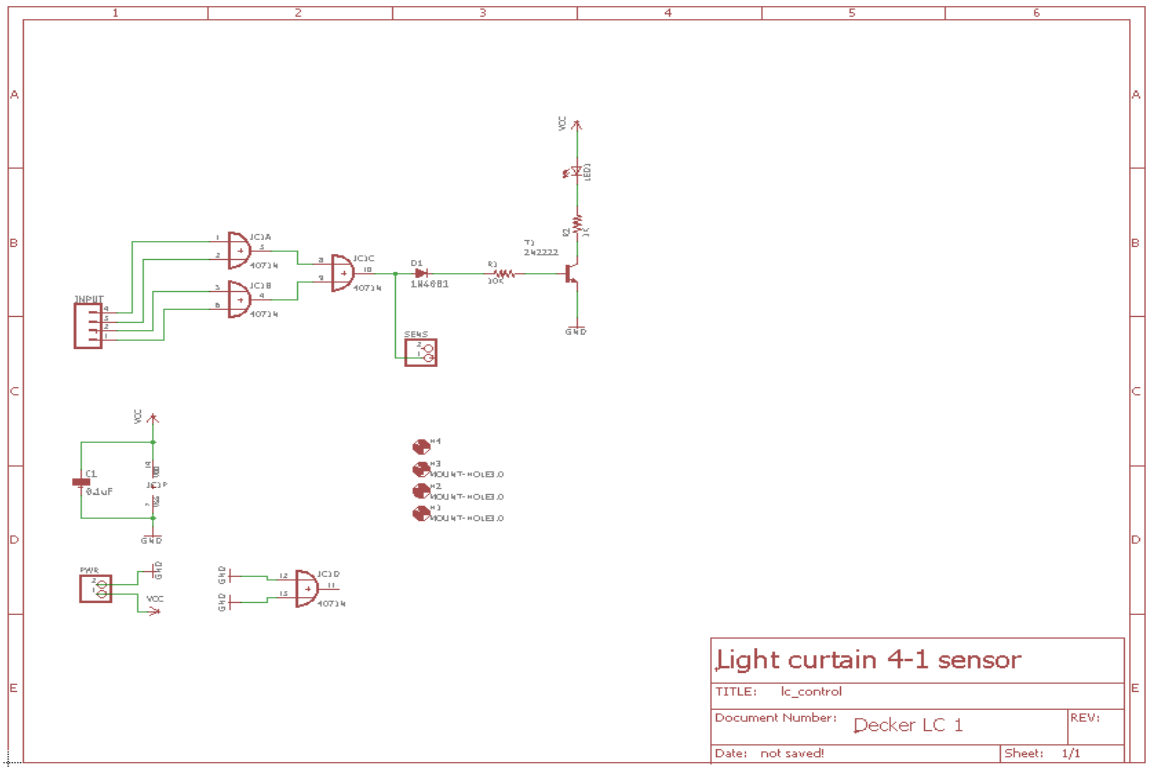


Figure 7 Light Curtain Control schematic (Brian L. Decker, 2018)

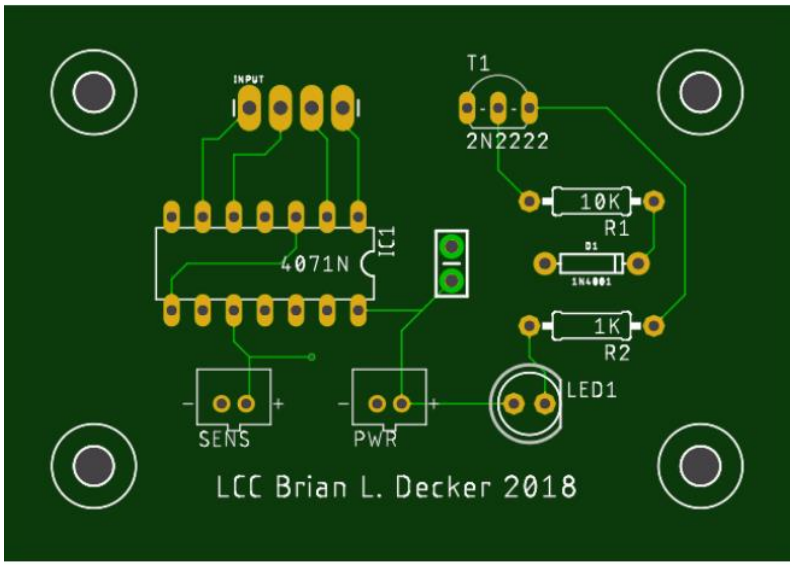


Figure 8 Light Curtain Control PCB – Gerber view (Brian L. Decker, 2018)

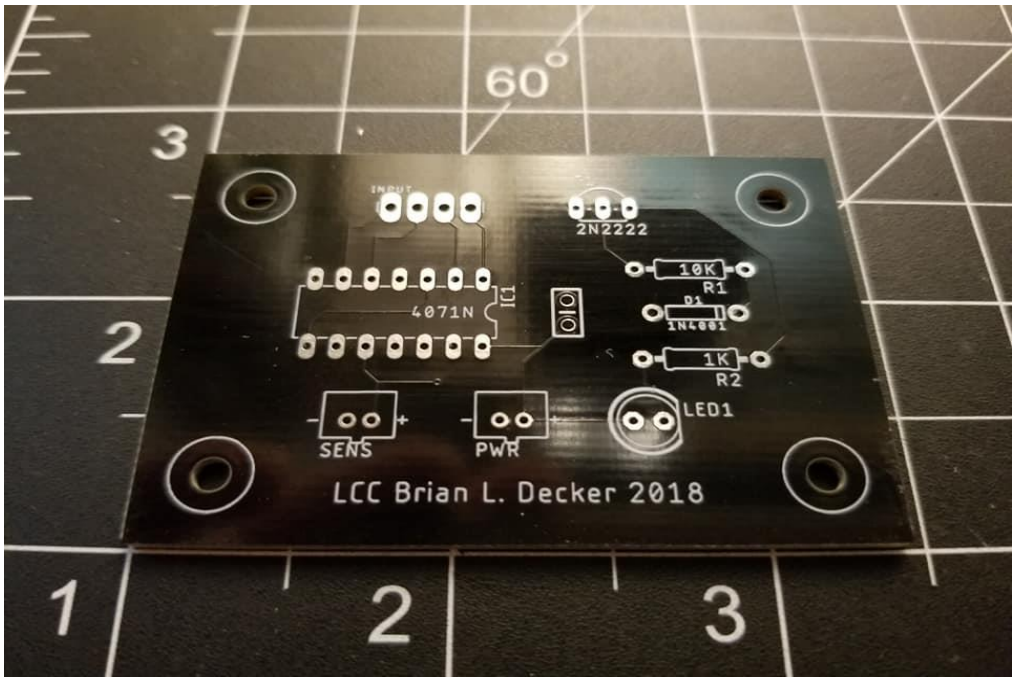


Figure 9 Light Curtain Control PCB (Brian L. Decker, 2018)

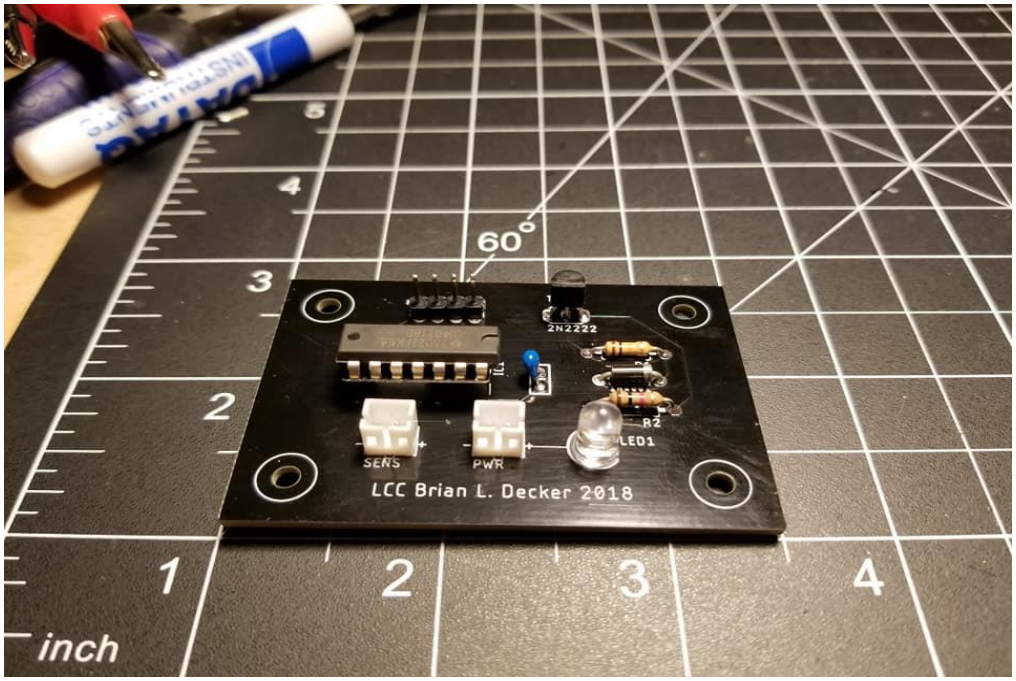


Figure 10 Light Curtain Control PCB – Assembled (Brian L. Decker, 2018)

SIM900 GSM module

The SIM900 module was developed by SIMCOM to provide an affordable solution for GSM communications of text and data. By utilizing the global 2G data network and a quad band (850, 900, 1800 and 1900 MHZ) transceiver the SIM900 can be easily implemented globally and tied to any cell provider. For the purposes of the L.M.C.S., I utilized a sim card from "Ting" as it allows for inexpensive monthly rates and unlimited texting. The phone number associated with the sim card provides a point of accessibility and differentiation from other devices in use. Because the L.M.C.S. leverages the global GSM network and quad band transmission, its hardware can be easily implemented anywhere within the 90% of the globe that is covered by 2G. Unlike devices that rely on local radionic networks such as Bluetooth or Wi-fi, there are none of the distance, network or security concerns that accompany those solutions.

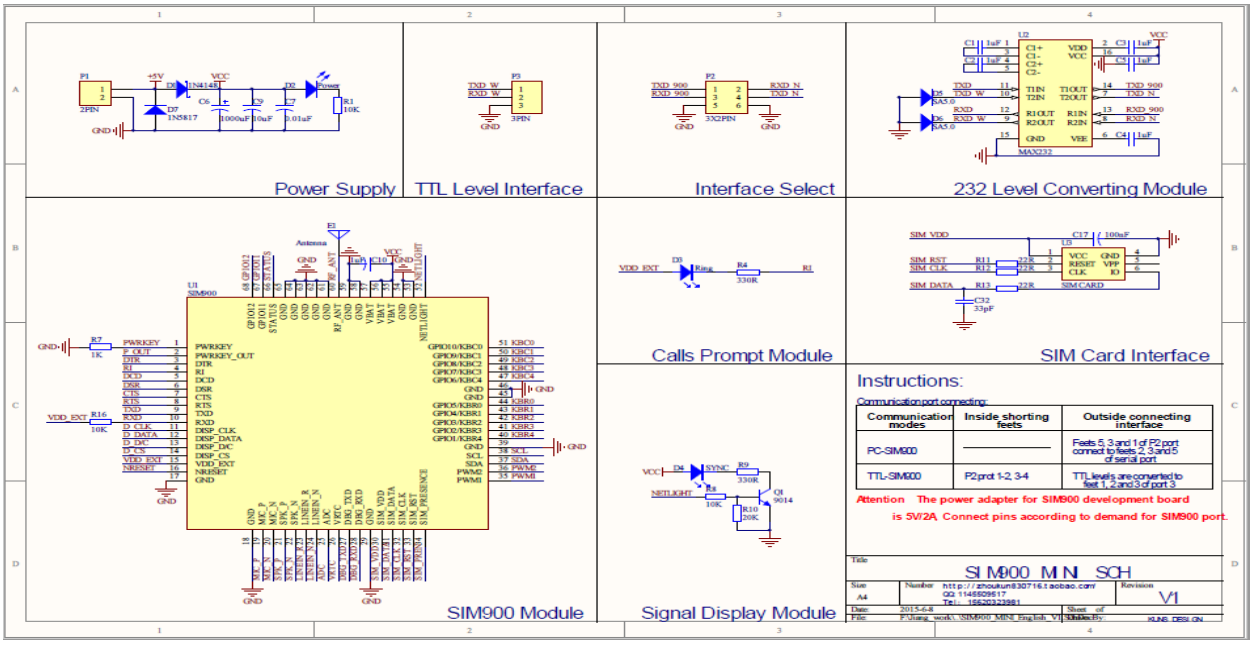


Figure 11 SIM900 GSM Module (SIMCOM, 2018)

OSEPP light sensor module

One of the environmental and security criteria tracked via the L.M.C.S. is the light level status of the manufacturing lab. The light level is obtained via the use of a Cds Photoresistor which generates an analog signal between 0v and 5v depending on the strength of the light it is exposed to. By processing this voltage via the ATmega 1284 ADC (analog to digital converter) it is possible to assign values representing both basic light conditions – "On" or "Off" – as well as lighting gradients "Dim", "Bright", etc. The ATmega 1284 can differentiate voltages to within 1/2048th volt – data points representing changes in light far to discreet for the human eye to make a meaningful determination. Currently, the light sensor is configured for a binary input outputting "On or "Off" based on the voltage output level. I coded a "dead- band" into the ADC to prevent intermediate lighting conditions from forcing high frequency updates to the microcontroller. In the future, I would like to expand this functionality to allow for discreet illumination levels to both be queried via text and used for output controls directly.

Figure 12 Light Sensor module (OSEPP, 2018)

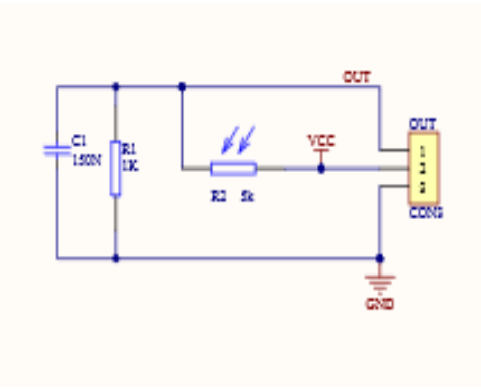


Figure 13 Light Sensor module schematic (OSEPP,2018)

DHT-11 Temperature and Humidity Sensor

The two other environmental sensors – for temperature and humidity levels – are combined into one module. The DHT-11 utilizes solid state sensors and an onboard ADC to interpret the signal from the sensors and transmit a single digital signal to the L.M.C.S. main controller which then parses the data and stores it as a sting in the text buffer. Each sample is compared to the text buffer, and only updated when a change in parameter is received. To avoid errors to temperature and/or humidity levels due to heat generated by the power supply, the DHT-11 module is mounted externally on top of the L.M.C.S. enclosure and connected to the main PCB via a 3-line ribbon cable providing VCC, GND and Signal paths. The three environmental parameters can be viewed directly on the L.M.C.S. main unit display screen or queried via GSM by texting the command "ENV" to the L.M.C.S. – this will trigger the command language parsing function and return a text to the phone issuing the query which includes the current temperature, humidity level and lighting status.

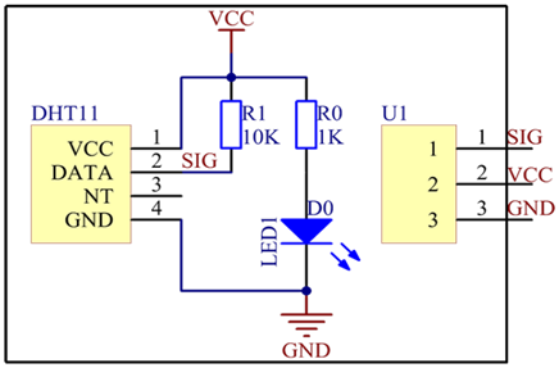


Figure 14 DHT-11 Schematic (OSEPP, 2018)

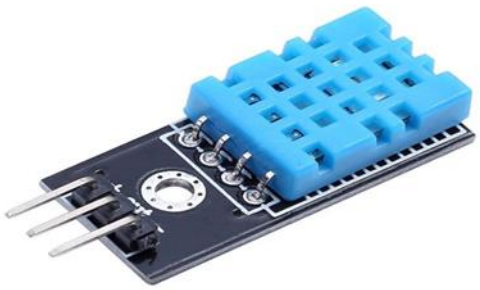


Figure 15 DHT-11 Assembled (OSEPP, 2018)

Project Outcomes: Function of the Product/Process

The project outcomes and measurable process milestones are as follows:

1. Produced a final schematic of the L.M.C.S. unit including all sub-modules.
2. Built a prototype "proof of concept" L.M.C.S. unit and performed electrical testing (both simulated and actual), "C" code optimization and performance testing to verify correct operation of the unit and conformity to specifications for operation.
3. Developed a bill of materials to include research of potential vendors and cost analysis that anticipates serial production to make best use of quantity discounts for common parts.
4. Created a PCB layout based on best practice manufacturing principles and provided the Gerber files to a board house for test unit manufacturing. Once produced, the PCB will be populated with appropriate components as determined during the BOM analysis and utilized as the final proof of concept unit in the NKU robotic lab.
5. Implemented concepts of math and science, engineering technology, practical techniques and competencies acquired within the EET program at NKU in the development of the L.M.C.S. unit as demonstrated by calculations using scientific laws, simulations using software, drawings, various production machines within NKU, or other documents.
6. Demonstrated ability with software specific to the production and manufacture of PCB based technology such as Autodesk Eagle, National Instruments Multisim and SEEED Fusion Gerber editors and viewers.

7. Wrote, debugged and implemented all “C” code required by the 8-bit AVR RISC-based microcontroller to perform the data processing, I/O functions, text-parsing and GSM interface functions required to meet the functions outlined in the proposal. All code will be written in the Microsoft Visual Studio 2017 IDE and the final operating code will be “burned” to the microcontroller via a custom Atmel boot-loading system.
8. Utilized industry standard development tools as provided by vendors and distributors – examples to include BOM management tools from Mouser and/or Digikey, data sheet compilers from Texas Instruments and code analysis tools from Microsoft and/or Atmel.

Assessment Method for evaluating Achievement of the Project Outcomes

The prototype unit will be constructed on a solderless breadboard to allow for electrical measurements – power consumption, electrical interference, power supply stability – are within design parameters. This will also allow for easy programming and code adjustments to the micro-controller itself as it may be directly connected to a computer via a FTDI programming header. Once correct operation and stability is achieved, the code base will be locked, and a final version of the operating software will be burned to the micro-controller. A custom PCB will be designed and laid out (utilizing Autodesk Eagle) and the Gerber files sent to a board house for manufacture of the final PCB. The project will then be assembled using the PCB and components identified as the most cost effective by comparative analysis. Once the board is complete it will be placed in the final enclosure and mounted near the robot safety cage in the NKU manufacturing lab, connected to the presence sensors and all operation parameters will be tested according to a

test plan. This plan is to include all local and remote I/O functions including GSM inquiries and control. The overall project assessment will be based on the results of this testing and the ability of the L.M.C.S. to meet the objectives outlined above. While providing a secure state for the robot safety cage is the primary objective of the project, the successful development and implementation of the text-parsing control system presents both the greatest challenge and defines my ability as an EET student.

Components of the Product/Process

The chart below indicates general prices that have not been evaluated for volume discounts or shipping costs. The final cost per item will be determined during the course of bill of materials analysis during the semester.

Part	Value	Price
ADMIN	LED	$ 0.18
ARMED	LED	$ 0.18
BTN_INC	Momentary Switch (Pushbutton) - SPST	$ 0.23
BTN_MEM	Momentary Switch (Pushbutton) - SPST	$ 0.23
BTN_RES	Momentary Switch (Pushbutton) - SPST	$ 0.23
BTN_SEL	Momentary Switch (Pushbutton) - SPST	$ 0.23
C1	22pF ceramic capacitors	$ 0.19
C2	22pF ceramic capacitors	$ 0.19
C3	0.1µF ceramic capacitors	$ 0.19
CONN_SENSOR_A	1x2 Header	$ 0.35
CONN_SENSOR_B	1x2 Header	$ 0.35
D1	1N4004	$ 0.08
DATA	LED	$ 0.18
DISARMED	LED	$ 0.18

Part	Value	Price
FTDI_HEADER	PROGRAMMING HEADER	$ 2.49
IC1	Atmel ATmega1284P 8-bit AVR Microcontroller	$ 5.13
ICCAP	10uF	$ 0.62
J2	1X5	$ 0.78
LS1	BUZZER-PTH	$ 0.95
OLED_DISPLAY	OLED	$ 2.80
OVER-RIDE	LED	$ 0.18
POWER	LED	$ 0.18
R1	10k	$ 0.09
R2	1k	$ 0.09
R3	10k	$ 0.09
R4	10k	$ 0.09
R5	10k	$ 0.09
R6	1k	$ 0.09
R7	1k	$ 0.09
R8	1k	$ 0.09
R9	1k	$ 0.09
R10	1k	$ 0.09
R11	1k	$ 0.09
R12	1k	$ 0.09
R13	220	$ 0.09
R14	10k	$ 0.09
R15	10k	$ 0.09
R16	10k	$ 0.09
R17	10k	$ 0.09
R18	10k	$ 0.09
R19	10k	$ 0.09
SENSOR_A	LED	$ 0.18
SENSOR_B	LED	$ 0.18
SW_ADMIN	SLIDING SWITCH	$ 0.86

Part	Value	Price
SW_ARMING	SLIDING SWITCH	$ 0.86
SW_RESET	Momentary Switch (Pushbutton) - SPST	$ 0.23
SW_SIGNAL	SLIDING SWITCH	$ 0.86
SW_TEMP	SLIDING SWITCH	$ 0.86
U$1	SW420 MODULE	$ 4.25
U$2	OSEPP	$ 3.95
Y1	16MHz Crystal	$ 0.49
SIM-900	GSM Module	$ 20.99
	Total of component parts (minus enclosure)	**$ 51.54**

Summary Cost Analysis

The L.M.C.S. utilizes standard electronic components available from any electronic parts supplier, however the PCBs are proprietary, and both the schematic design and PCB layout were created utilizing Autodesk EAGLE. The parts and assembly cost could be further reduced by replacing many of the Part Through Hole (PTH) parts with Surface Mount Devices (SMD) – however, due to limited experience with SMD components it was decided that PTH components should be used to assure successful completion of the project.

PART	COST
LED	$1.50
SPST Slide	$3.25
SPST Momentary	$2.00
Capacitors	$3.23
Prog Header	$1.63
FTDI - header	$4.10
1N4004	$0.10

- The overall parts cost per unit: $73.24
- The per unit cost including assembly is: $123.24
- A typical MSRP may be determined by using the formula (Unit Cost*2.75) resulting in a unit sale price of: ~$350.00
- Because the L.M.C.S. utilizes a cellular

Atmega 1284	$5.13
OLED	$2.83
Buzzer	$0.78
Resistors	$4.25
SW-420	$3.28
16Mhz crystal	$0.49
SIM-900	$20.99
Type 86 Case	$4.68
Shipping	$15.00
parts subtotal	$73.24
assembly (2hrs)	$50.00
Unit Cost	$123.24
MSRP	$338.91
TING Sim (month)	$9.00

connection, there is an ongoing SIM cost of $9.00 per month. If commercialized, the MSRP cost of a hosted service would be (9*2.75): $24.75/month

- The resulting revenue per unit sold at MSRP with hosting would result in: $225+$189/year
- The required units to reach a one million dollar ROI (including PCB design and manufacture) is: ~5,000

Design Parameters, Testing, Commercialization

Design criteria: The design is to adhere to the parameters as established by the component manufacturers in the relevant datasheets or installation guides.

Constraints: The primary constraints are (A) time to develop, debug and install the completed L.M.C.S. unit and (B) funds available for the purchase of required components.

Calculations: All relevant calculations relating to electrical requirements, system performance and individual component needs will be collected into the final project report as an appendix.

Materials/process selection: Materials will be selected according to their suitability to the product and consideration of their cost/quality parameters. In cases where such determination is unclear, further feedback will be sought from supervising faculty.

Cost Analysis: Will be performed by comparisons of component acquisition cost (price + shipping)

among several suppliers. This data will be analyzed via Excel.

Time management: Similar to the cost analysis, the time management process for the project will consist of an Excel based spreadsheet and hours will be quantified against standard wages to determine a labor cost for design, production and testing costs.

Assembly: Will be performed according to the specifications set forth by the component manufacturers in a workspace that provides for appropriate ventilation, contaminant controls and ESD protection.

Actuation: Once installed, the operation of the L.M.C.S. unit is automatic and requires user input only when an operational state change is required. The L.M.C.S. unit is intended to operate continuously. Remote GSM inquiries for status or control input will be generated automatically via the text parsing system.

Testing of Product/Process: The L.M.C.S. will be tested according to a test plan to include all I/O functions including GSM inquiries and control. This testing will initially be conducted on the prototype system – once completed successfully, the initial version of the L.M.C.S. (including the custom PCB) will be tested according to the same test plan. The unit will then be installed in the lab and a final test plan will be conducted which includes verifying the operation of all I/O functions and sensor systems in a functional environment.

Commercialization: The design of the L.M.C.S. unit is intended to provide for serial production of the unit once testing has been completed. Each step of the development process will be documented in a way that supports the provision for future manufacturing without additional changes or refinements. The project is not only intended to produce a functional control system for use in the NKU lab, but one which may be produced and distributed as a commercial product.

Competencies Demonstrated

EGT 116: Metal forming techniques and procedures.

EGT 161: Basic laws and theories, voltages, current, power, and resistance; resistive circuits in direct current circuits; analysis and applications.

EGT 212: production of technical drawings using CAD software.

EGT 243: Application of basic electrical circuit analysis to alternating current systems; capacitors, inductors, transformers, and circuits using these components.

EGT 245: Digital circuits; logic, registers, counters, arithmetic circuits, and memories.

EGT 260: Relevant OSHA standards and practices relating to robotic safety cage requirements.

EGT 267: Engineering programming using Integrated Development Environment.

EGT 310: Industrial project management practices: project implementation methods; resource selection; risks and failures; project management tools and techniques related to manufacturing projects.

EGT 321: The integration of systems required to improve work flow through the system, scheduling and coordination of projects.

EGT 344: Use of solid state, diodes, bipolar and field-effect transistors, small-signal amplifiers; power amplifiers, voltage regulators, and active filters.

EGT 365: Knowledge needed to set up and program Computer Numerical Control (CNC) machines equipped with EIA or Conversational programming formats.

EGT 367: Architecture, instruction sets; programming, interfacing, and designing with microprocessors.

EGT 386: Introduction to the design of instrumentation and control systems. Study of thermal, mechanical, optical, and digital sensor operations and applications. Introduction to data acquisition systems. Laboratory experimentation involving the programmable logic controls for designing different logics to control devices and selecting sensors to gather and utilize data from the equipment at hand.

EGT 411: Organizing and implementing the quality audit; types of audits; ISO 9000 quality standards; audit planning, execution, testing, reliability, and system appraisal.

EGT 412: Advanced features of three dimensional and parametric modeling.

EGT 462: Fundamentals of finite element modeling, creation of geometry, material selection, and problem solving.

EGT 467: Advanced architectures, multitasking, virtual memory, networking, assembly language.

Future Work

The core of the L.M.C.S. consists of two technologies – GSM communication and a text parsing algorithm – that are combined to allow for digital information to be transmitted in a human readable form. In this implementation, that combined technology is being used to monitor the robot safety cage and general manufacturing laboratory environment; however, with a simple change in sensors the same controller could be used to monitor material levels in a storage tank, critical weather data from a remote station or the presence of contaminants in a watershed. That data would then be transmitted to one or more designated devices anywhere in the world via GSM over a cellular network that covers 90% of the occupied land surface of the globe. Working toward this development largely represents the next steps we hope to take.

Functional Block Diagram

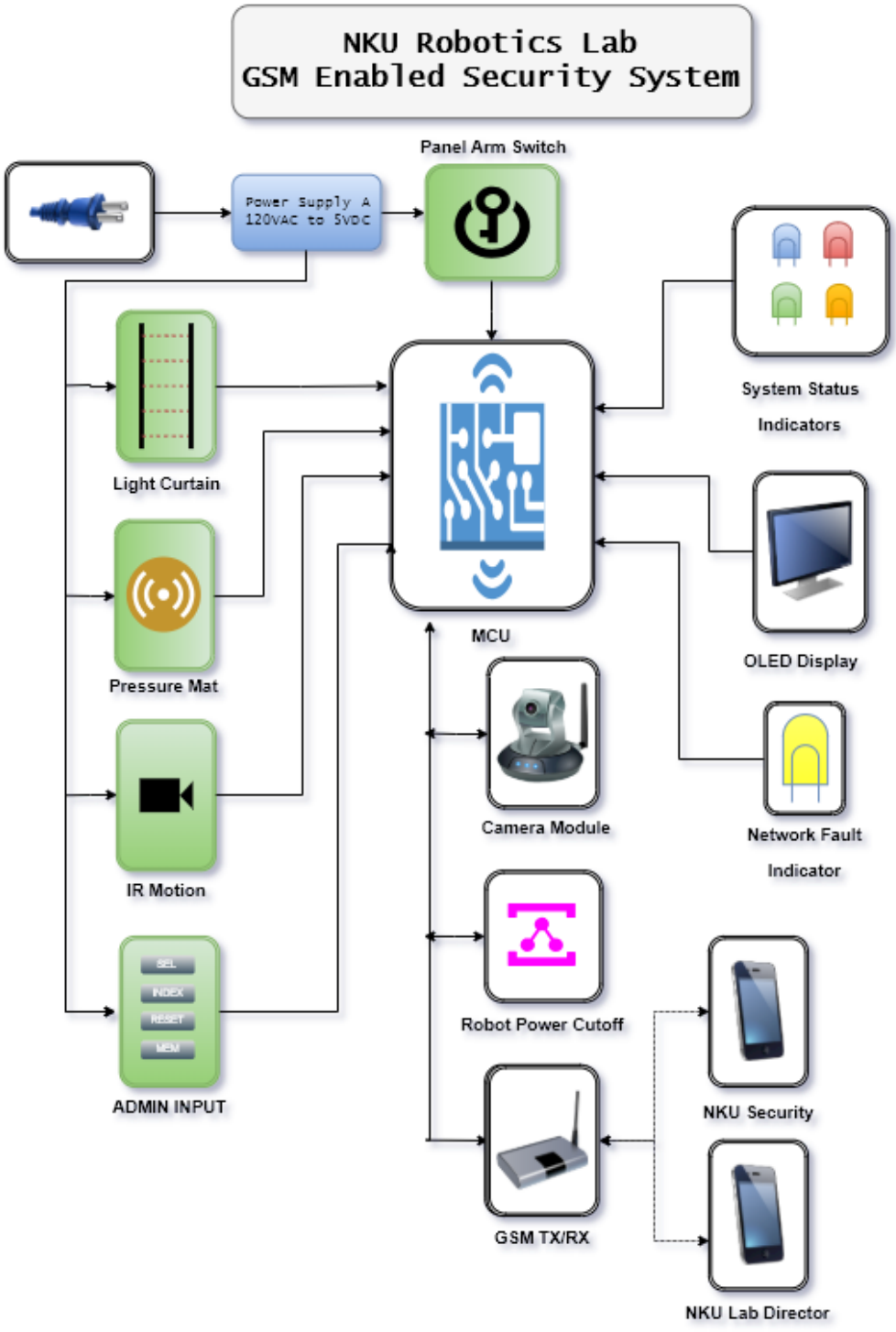
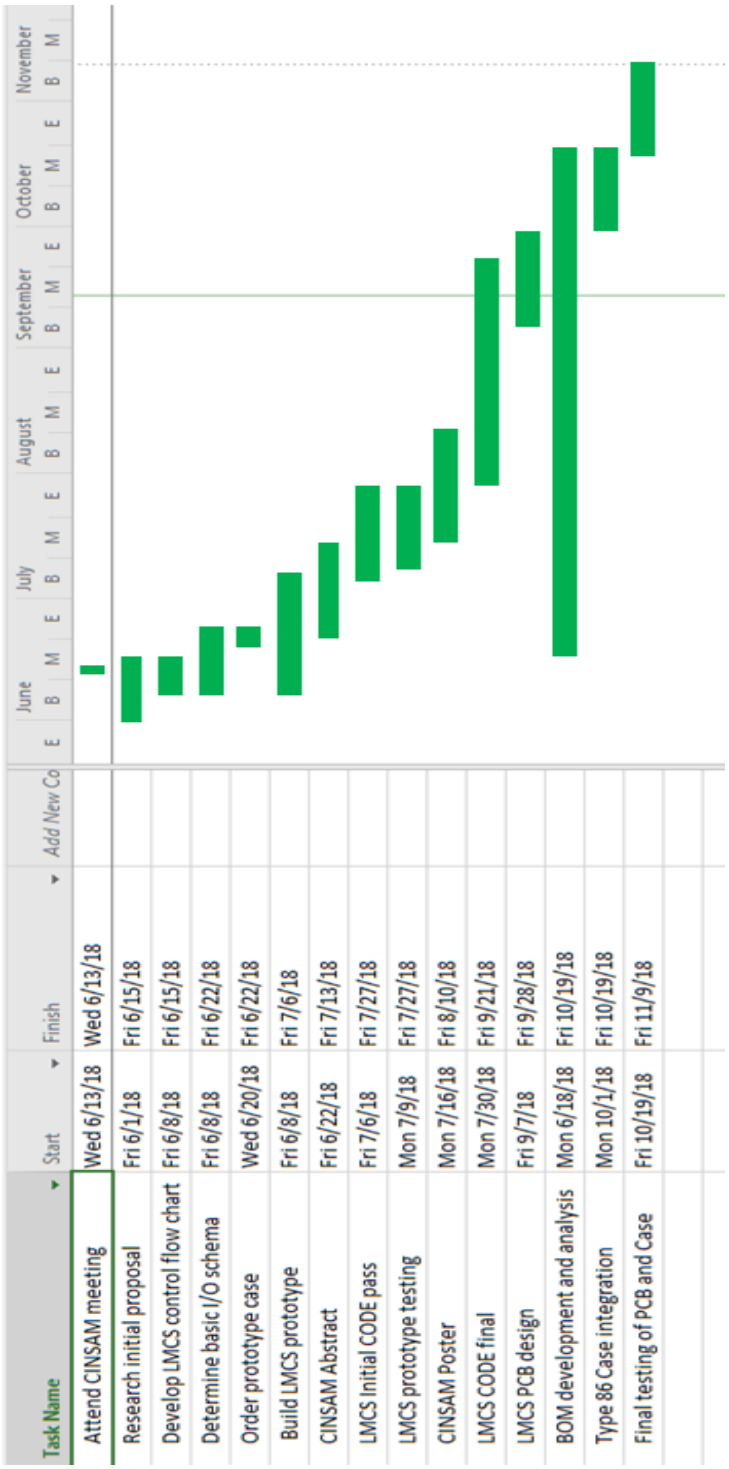


Figure 16 Functional Diagram (Brian L. Decker, 2018)

Gantt Chart of Overall Development



Task Name	Start	Finish	Add New Co
Attend CINSAM meeting	Wed 6/13/18	Wed 6/13/18	
Research initial proposal	Fri 6/1/18	Fri 6/15/18	
Develop LMCS control flow chart	Fri 6/8/18	Fri 6/15/18	
Determine basic I/O schema	Fri 6/8/18	Fri 6/22/18	
Order prototype case	Wed 6/20/18	Fri 6/22/18	
Build LMCS prototype	Fri 6/8/18	Fri 7/6/18	
CINSAM Abstract	Fri 6/22/18	Fri 7/13/18	
LMCS Initial CODE pass	Fri 7/6/18	Fri 7/27/18	
LMCS prototype testing	Mon 7/9/18	Fri 7/27/18	
CINSAM Poster	Mon 7/16/18	Fri 8/10/18	
LMCS CODE final	Mon 7/30/18	Fri 9/21/18	
LMCS PCB design	Fri 9/7/18	Fri 9/28/18	
BOM development and analysis	Mon 6/18/18	Fri 10/19/18	
Type 86 Case integration	Mon 10/1/18	Fri 10/19/18	
Final testing of PCB and Case	Fri 10/19/18	Fri 11/9/18	

Figure 17 Gantt chart of L.M.C.S Project (Brian L. Decker, 2018)

Appendix A: Bi-Weekly Updates

Week Ending:	8/24/2018
Student Name:	Brian L. Decker
Hours Worked on the project	20
Faculty Advisor	Dr. Gang Sun

Bi-Weekly Summary: Please answer the following questions

1. *What were Objectives and Goals of the past week?*

My primary focus for the past week has been two-fold:

I. Completion of the presentation documentation for the Heather Bullen Summer Research Celebration.

II. Prepare the L.M.C.S. prototype unit for display at the Heather Bullen Summer Research Celebration.

2. *How did you achieve the above objectives/goals?*

I. I met with Dr. Kulkarni to verify the set-up time for the display at the celebration and picked up the poster from the CINSAM office:

Laboratory Environment Monitoring, Safety and Control System with GSM Interface

Brian L. Decker

Faculty Mentors: Dr. Gang Sun, Dr. Mauricio Torres – Department of Engineering Technology

Introduction

The expectations of the Laboratory Environment Monitoring, Safety and Control System (LMCS) for the NKU robotic systems lab are threefold:

I. Provide the OSHA required safeguard controller to prevent movement of the robotic system while a person is present within the safety cage area.
II. Allow for real-time remote monitoring of the environmental conditions in the NKU lab such as temperature, humidity and light levels. These are often important factors in the successful application of modern manufacturing techniques when working with specialized coatings or curing composite materials.
III. Send notifications via text message to the lab manager, engineering faculty and designated students if the current laboratory environmental variables exceed pre-set parameters. Additionally, NKU security and designated faculty will receive immediate notification of individuals violating safety cage protocol.

Communication

Figure 1 Cincinnati Area GSM Coverage (T-Mobile, 2018)

The LMCS is unique in the use of a GSM plain-text parsing system for both input and output functions. This system does not require the use of external applications, is secured via encryption and access or control can be reconfigured to meet the changing needs of students, security and faculty.

Hardware Development

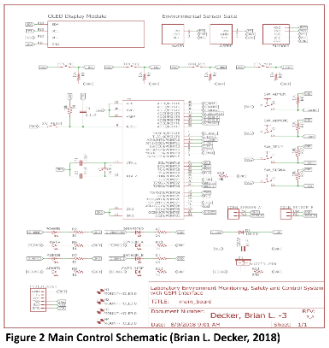


Figure 2 Main Control Schematic (Brian L. Decker, 2018)

The LMCS will use the powerful Atmega 1284 IC as its CPU. This 8-bit AVR RISC-based microcontroller will process the input from the user interface controls, presence and environmental sensors and GSM text parsing system while providing output to the OLED display screen, LED indicators, tone generator and text alert system.

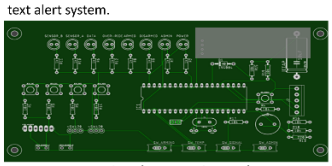


Figure 3 Main Control PCB (Brian L. Decker, 2018)

Software Development

Figure 4 Visual Studio IDE (Microsoft, 2018)

The operating software of the LMCS was developed using the Microsoft Visual Studio IDE, and consists of over 2500 lines of "C" code. The software integrates all aspects of the hardware system: sensors, sounds, OLED display, LED indicators, user interface controls and the GSM communication platform. The LMCS is designed for unattended operation, and the software utilizes both self loading and error checking routines to allow for restarts in the event of a power interruption or sensor failure. Should a system error be detected, the LMCS attempt to resolve the error via an automated restart. The LMCS software supports the following four modes of operation:

I. Armed: Presence and environmental sensors selected ON and text parsing system is active.
II. Disarmed: Only Environmental sensors selected ON and text parsing system is active.
III. Run: Only Presence sensors selected ON and text parsing system is inactive.
IV. Admin: While in this mode the LMCS text contact numbers may be edited or updated, OLED display settings changed or the environmental condition alert parameters may be adjusted via the user interface buttons on the front of the system enclosure.

NKU

Figure 18 PGET-04 CINSAM Poster (Brian L. Decker 2018)

II. I rebuilt the L.M.C.S. prototype circuit from a basic breadboard layout to a custom multi-row solderless protoboard that was attached to a powder coated metal backing plate. I cut two pieces of hardwood 1x1 material and attached the protoboard to the wood to provide proper positioning.

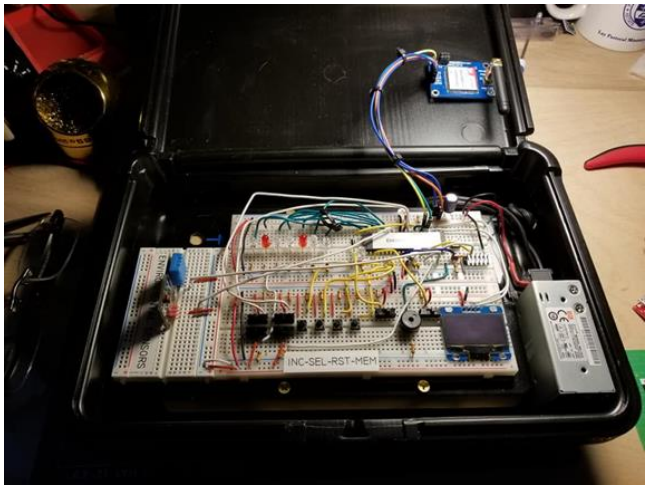


Figure 19 L.M.C.S. prototype in temporary case (Brian L. Decker 2018)

This assembly was then mounted in a generic plastic carrying case and secured temporarily with small pieces of double sided foam tape. I drilled a hole in the side of the case and after wiring a MW RS-15-5 2 amp switching power supply to the protoboard I secured the 110-volt power cord and verified ground continuity and proper operation.

Having completed the protoboard and verified power and ground continuity, I tested the L.M.C.S. functions including the power supply, GSM module, OLED and indicator LED's. I then used a Tektronix 2235A oscilloscope to measure the output signal strength from the SIM-9000 with an external antenna connected instead of the internal unit that can often be too weak to reliable transmit from within a building.

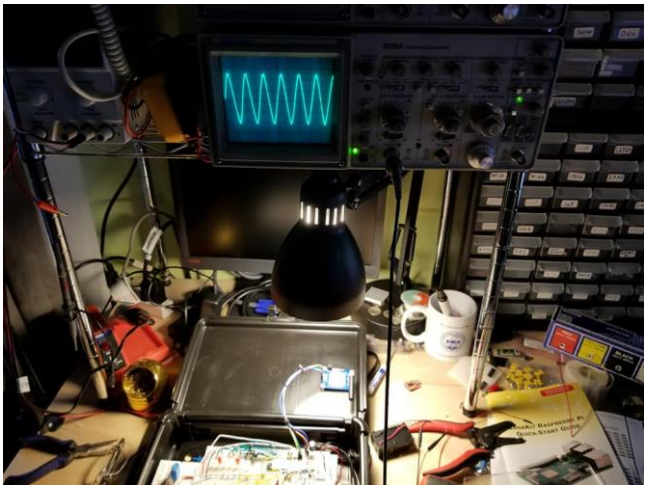


Figure 20 SIM-9000 antennae test setup (Brian L. Decker, 2018)

Week Ending:	9/14/2018
Student Name:	Brian L. Decker
Hours Worked on the project	42.5
Faculty Advisor	Dr. Gang Sun

Bi-Weekly Summary: Please answer the following questions

3. *What were Objectives and Goals of the past week?*
   The primary objectives for the last two weeks:

   I. Finalize the code that allows phone numbers and other input to be changed and stored to the EEPROM.
   II. Explore the use of a RTC and associated code so that time displays on the OLED and may be used as a parameter to determine the appropriate text message alerts to be sent and to which phone.
   III. Begin work on the "Walking Trail People Counter" project assigned by Dr. Sun per the request of Dr. Durtsche.

4. *How did you achieve the above objectives/goals?*

III. Dr. Sun has limited the number of physical input controls for the L.M.C.S. to 4 momentary pushbuttons and a multi-position key switch. This has created an interesting problem to solve: How do you enter a 10-digit phone number with only 4 buttons?

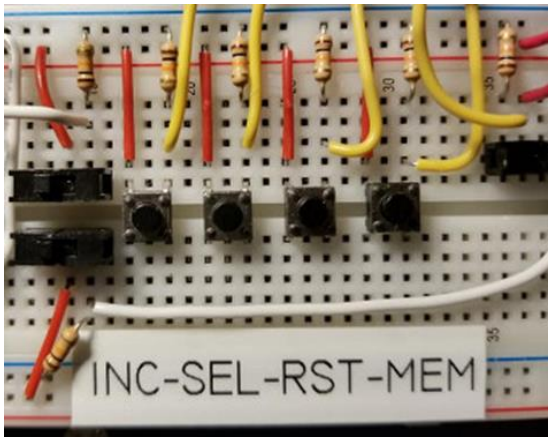


Figure 21 L.M.C.S. 4-Button Input (Brian L. Decker, 2018)

Obviously having phone numbers hard coded to the L.M.C.S. would work for a period of time, but it is far from an optimal solution as students, faculty and others who may want to interact with the L.M.C.S. will require that the phone number stored in the system be updatable. Additionally, there are other parameters that will require changes – setting a temperature, humidity or light value warning or setting the time.

My solution is to use an array matrix that stores each phone number and each possible digit (0-9), a cursor position reference call and an EEPROM PUT statement to store the result. To enter a phone number, the L.M.C.S. is placed in ADMIN mode via the key switch.

Figure 22 L.M.C.S. OLED (Brian L. Decker, 2018)

The OLED display will show the current Phone #A stored in EEPROM via a GET function call and the cursor will be displayed under the first digit. Pressing the INC pushbutton will increment the number from its current value, rolling back to "0" after "9". Once the correct number is displayed, pressing the SEL pushbutton will move the cursor to the next number.

The array matrix uses this change in cursor position as a pointer and now pressing the INC button will change the 2nd number. This process is continued until all 10 digits have been updated.

Figure 23 L.M.C.S. OLED (Brian L. Decker, 2018)

If an error is made, pressing the RST will return the cursor to the first position. Once the number is entered correctly, pressing the MEM button will call an EEPROM PUT function and write the number from the array matrix into the L.M.C.S. EEPROM where it will remain even if power is lost or the system is restarted.

IV. I would like to display the time and date on the L.M.C.S. OLED display. The most reliable way to do this is to add an RTC with a small coin cell battery backup. I have assembled a generic RTC module and added to the L.M.C.S. prototype. I have the code complete for the basic display functions, however I am still working on the code that will allow for the time to be used as a parameter in the text notifications – i.e. NKU security should be notified if the cage is accessed outside of lab hours, but not during lab hours. I believe that this is a critical addition to the controller and will have the code finalized by early next week.

V. Dr. Sun asked that I re-work a project from a previous student that had been created in conjunction with Dr. Durtsche – a basic IR sensor and counting routine that will be placed near a walking trail to count the number of people who pass the sensor. We discussed the prototype nature of the unit and the basic expectations of the functions required. I will have more information on this project in my next report.

Week Ending:	9/27/2018
Student Name:	Brian L. Decker
Hours Worked on the project	38
Faculty Advisor	Dr. Gang Sun

Bi-Weekly Summary: Please answer the following questions

5. *What were Objectives and Goals of the past week?*
   The primary objectives for the last two weeks:

   IV. Make a final pass through the L.M.C.S. code base and lock final version.
   V. Start PCB layout for main and peripheral boards.
   VI. Submit sensor purchase order to Dr. Sun.
   VII. Continue work on the "Walking Trail People Counter" project assigned by Dr. Sun per the request of Dr. Durtsche.

6. *How did you achieve the above objectives/goals?*

VI. My preferred coding style is to use the "Verbose/Concise" writing format. My prototype code is not optimized and is heavily structured with functions contained to themselves, and features added on top of a stack that I know compiles. As I add each new feature, I maintain the ability to compile and test to ensure that the overall stack has not been inadvertently corrupted. Once all features have been added, I make consecutive optimization passes and restructure the code into a more optimal format that makes proper use of arrays, switch statements and libraries or header files when appropriate. This adds some time to the overall code development process, however because the stack is kept "clean" it is always possible to test and debug easily.

VII. Start PCB layout for main and peripheral boards.

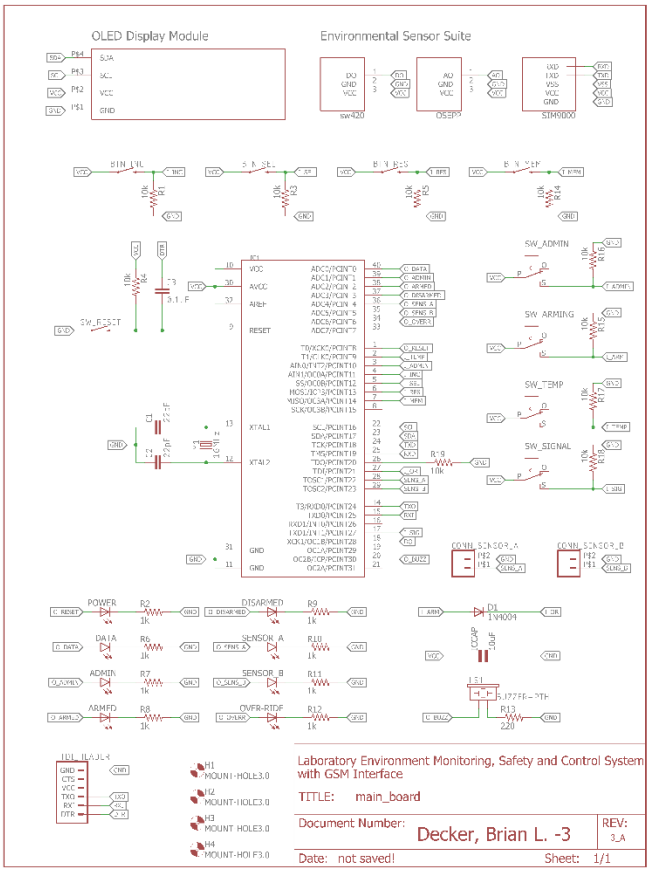


Figure 24 L.M.C.S. Prototype Schematic (Brian L, Decker, 2018)

I have a subscription to Autodesk EAGLE and have laid out the overall schematics for the L.M.C.S. based on the prototype layout. I will use this as the basis for the layout of the main and peripheral boards. Once complete, I will send the Gerber files to SEEED Fusion and have the initial prototype boards produced.

VIII. I have reviewed a variety of sensors for the doors of the robot safety cage and found a retroreflective IR sensor that meets the functional requirements and is cost effective. I submitted the PO to Dr. Sun on 9/26.

IX. I have deconstructed the parts provided by Dr. Sun for the trail counter project. My plan is to rebuild onto a soldered prototype board and forego the use of the solderless breadboard that had been used originally. I have communicated to Dr. Sun my concerns regarding the use of the PID sensor in place of a through beam, and he responded that because this is just a prototype the general trigger provided by the PID will be sufficient.

Week Ending:	10/12/2018
Student Name:	Brian L. Decker
Hours Worked on the project	42
Faculty Advisor	Dr. Gang Sun

Bi-Weekly Summary: Please answer the following questions

7. *What were Objectives and Goals of the past week?*
   The primary objectives for the last two weeks:

   VIII. Complete work on the "Walking Trail People Counter" project assigned by Dr. Sun per the request of Dr. Durtsche.
   IX. Continue PCB layout for main and peripheral boards.
   X. Finalize case for L.M.C.S.

8. *How did you achieve the above objectives/goals?*

X. I completed the code for the trail counter using the Visual Micro plug in for Visual Studio which allows for the use of standard "C" with an 8-bit RISC microcontroller like those made by Atmel. I am not a fan of the Arduino – which is what had been used by the previous student – and decided to replace it with a PCB I designed last year for my business (Encom Lab). It is based on the same ATmega 328 IC but uses a L7805 voltage regulator in place of the switching regulator on the Arduino and does away with the unnecessary components the Arduino requires for on-board programming such as a the UART controller and USB interface.

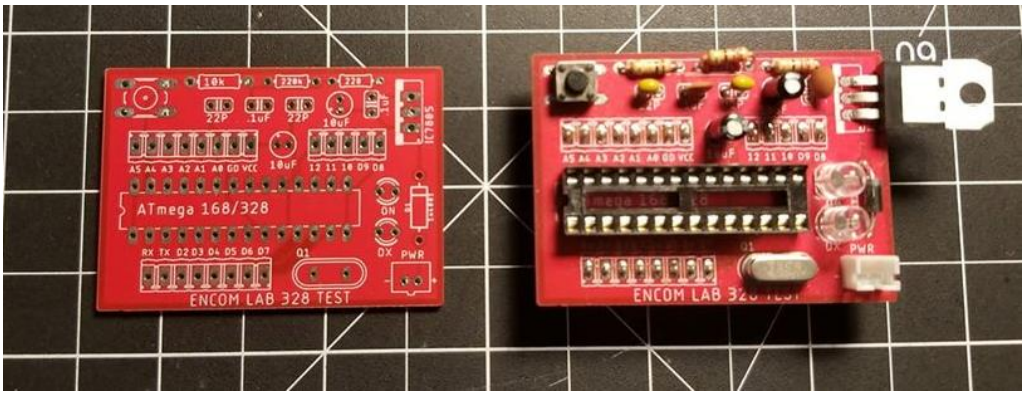


Figure 25 Encom Lab 328 Test PCB (Brian L. Decker, 2018)

I soldered the PTH components to the PCB and burned my code to an ATmega328 IC using a custom header interface and a 6-pin programming header

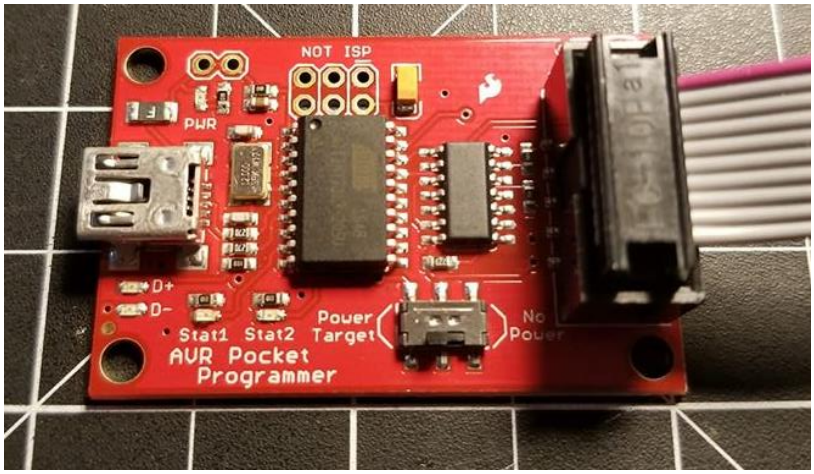


Figure 26 AVR Bootloader (Sparkfun, 2017)

Once the board was finished, I used a standard 100x60mm project case and machined mounting locations for the buttons, indicator LED and 2x16 LCD using a X/Y table mounted to my drill press.

Figure 27 X/Y table mounted to drill press (Brian L. Decker, 2018)

The end result is a compact unit with a simple UI and display.

Figure 28 Finished Trail Counter (Brian L. Decker, 2018)

XI. Continue PCB layout for main and peripheral boards.

I have a subscription to Autodesk EAGLE and have laid out the overall schematics for the L.M.C.S. based on the prototype layout. I will use this as the basis for the layout of the main and peripheral boards. Once complete, I will send the Gerber files to SEEED Fusion and have the initial prototype boards produced.

XII. Finalize case for L.M.C.S.

Using Autodesk FUSION 360 I mocked up a final case design for the L.M.C.S. This is intended only as a visualization, the final case will likely make use of an industry standard electronics isolation cabinet as that will utilize a dust gasket and possess standard mounting points and hardware.

Figure 29 L.M.C.S. 3d render (Autodesk Fusion, 2018)

Week Ending:	10/12/2018
Student Name:	Brian L. Decker
Hours Worked on the project	45
Faculty Advisor	Dr. Gang Sun

Bi-Weekly Summary: Please answer the following questions

9. *What were Objectives and Goals of the past week?*

The primary objectives for the last two weeks:

XI. Submit Gerber files to SEEED Fusion for PCB manufacturing

XII. Test continuity of pad connections and verify trace routing

XIII. Solder components to L.M.C.S. PCB, flash programming to ATmega 1284 and test functionality and system integrity

XIV. Machine openings in face plate of enclosure and begin wiring LED indicators, switches and key lock

10. *How did you achieve the above objectives/goals?*

XIII. Very big steps accomplished over the last two weeks as the Gerber files I produced from the PCB layout in Autodesk Eagle were sent to SEEED Fusion in Shenzhen for manufacturing. I was able to keep the finished dimensions of the main PCB to less than 100mm*100mm which qualified the board for the "prototype pricing" provided by SF.

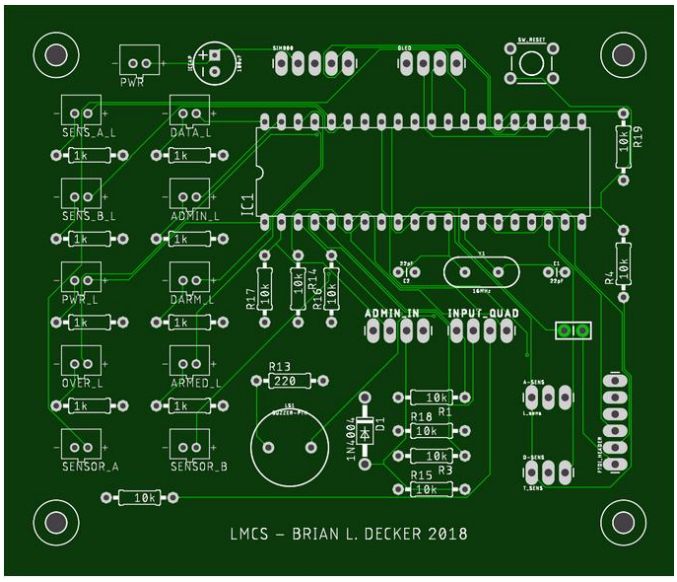


Figure 30 Final L.M.C.S. PCB layout Gerber view (Brian L. Decker, 2018)

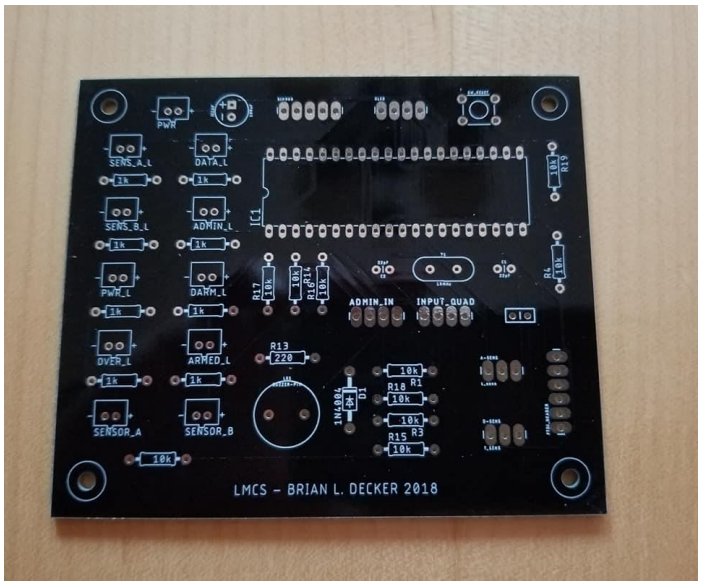


Figure 31 L.M.C.S. PCB from initial manufacturing run (Brian L. Decker, 2018)

XIV. I've been using SEEED Fusion board services for several years and am amazed that they can produce boards of this quality at the prices they offer. The cost (minus DHL shipping) for ten prototype boards was less than $5! As always, the first step in testing any new board design is a continuity check of all pads and traces against the specifications of the schematic. In the "real world" a bed of nails test rig would be configured for the board and all points tested at once – in my world, testing the board meant making a spreadsheet of all connections and settling in my favorite Fluke 115 multi-meter. Thankfully, all traces and pads tested properly and there were no dead pads or improperly routed traces.

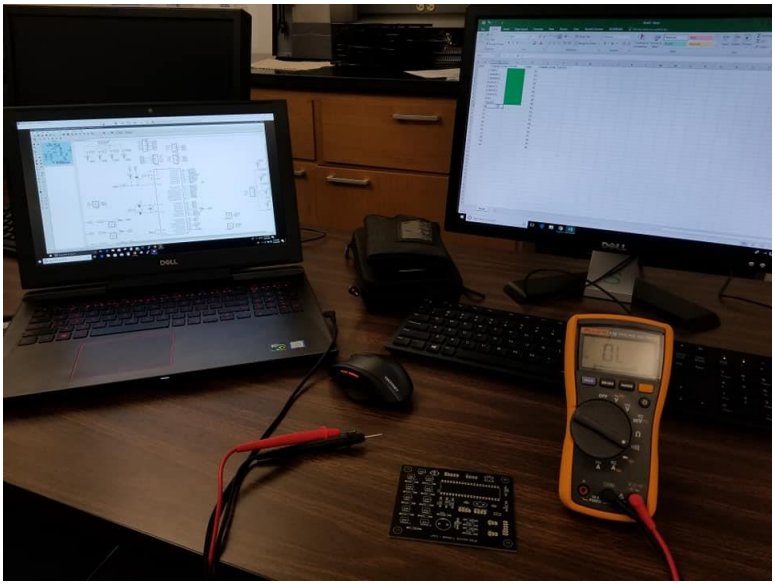


Figure 32 L.M.C.S. testing pad continuity and trace routing (Brian L. Decker, 2018)

XV. Once PCB verification was completed it was time to warm up my soldering station and start assembling. I soldered all components using a Hakko FX-888D soldering station and Loctite 0.5mm 60EN multicore solder.

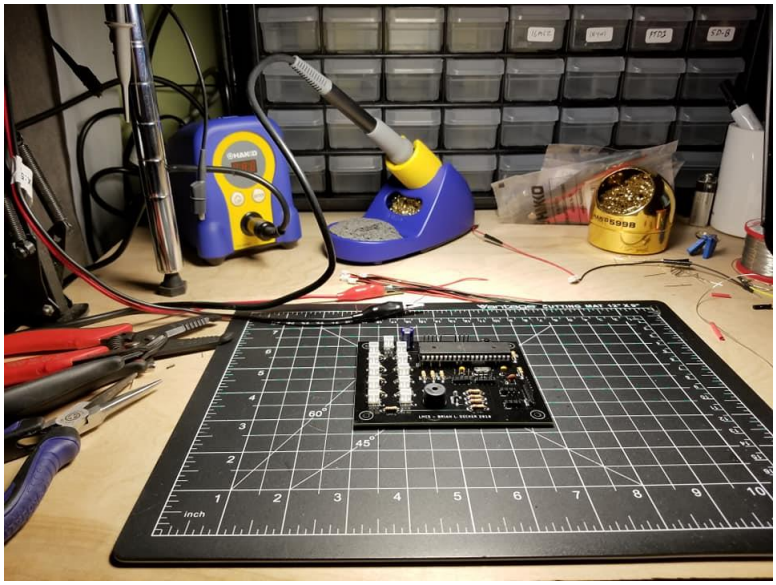


Figure 33 L.M.C.S. PCB assembled (Brian L. Decker, 2018)

The next step was applying power and verifying that the boot routine of the microcontroller initialized properly and that there were no obvious signs of solder bridges or misplaced components (i.e. magic smoke, sparks, fire and chaos). Fortunately, the boot routine ran fine, and the chip powered on as expected. I will be conducting a full test routine over the next two weeks once the face plate is finished.

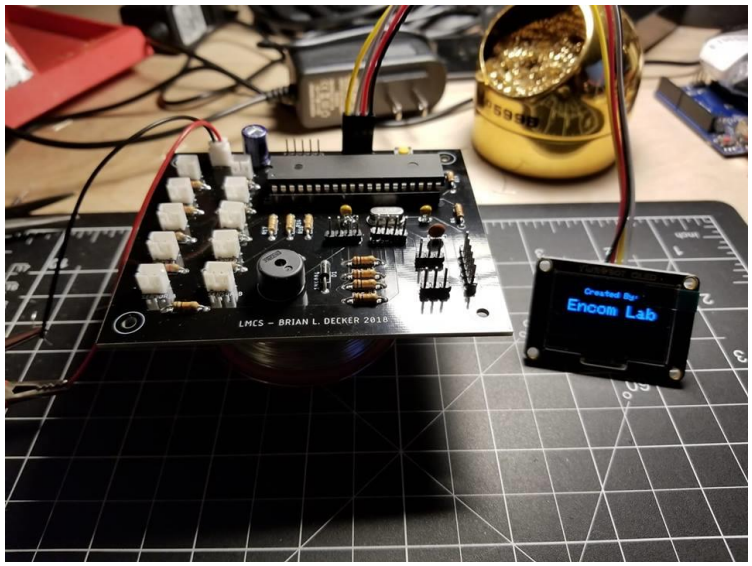


Figure 34 L.M.C.S. initial POST and boot sequence (Brian L. Decker, 2018)

XVI. Following the steps that I practiced while making the trail counter enclosure for Dr. Sun, I again used my X/Y table and mill/drill to cut the openings required for the OLED display mount, input pushbuttons and key lock selector switch into the faceplate of the L.M.C.S. test enclosure. The use of the locking leadscrews allowed me to align the LED holders and pushbuttons along a common axis, making for a more professional result. Once the locations had been determined, I used a low speed setting and slow feed to cut the openings from the ABS faceplate. Following cutting, I used a hot wire to smooth the openings and mounted the chromed LED holders, momentary NO pushbuttons and the three-position locking key switch. The last assembly step will be soldering the light and switch leads to their respective connections, crimping on JST-PH plugs and testing the completed unit.

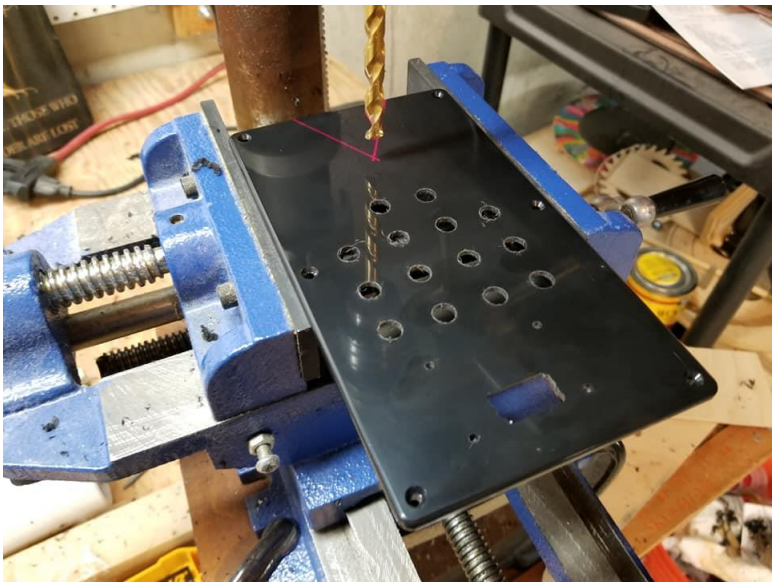


Figure 35 L.M.C.S. faceplate cutting (Brian L. Decker, 2018)

Figure 36 L.M.C.S. faceplate all holes complete and trimmed (Brian L. Decker, 2018)

Figure 37 L.M.C.S. test case completed (Brian L. Decker, 2018)

Appendix B: Operating System Code

L.M.C.S. SOURCE CODE

```
/* LAB MONITOR SYSTEM
   Created by: Brian L. Decker
   COPYRIGHT (C) 2018 Encom Lab.
   All rights reserved.
*/
#include <avr/wdt.h> //handles watchdog
//++++++++++++++++++++
//LIBRARY FILES

// oled library
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SH1106.h>

//eeprom library
#include <EEPROM.h>

//temp and humidity sensor library
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>
#define DHTTYPE           DHT11     //type definition

//DHT sensor pin
const int DHTPIN = 12;  //required here to satisfy next statement
DHT_Unified dht(DHTPIN, DHTTYPE);   //temp sensor type declaration

// display reset
#define OLED_RESET -1
Adafruit_SH1106 display(OLED_RESET);

// sim900 function calls via softserial
#include <SoftwareSerial.h>

// softserial inputs - tx, rx
SoftwareSerial SIM900(18, 19);

//++++++++++++++++++++
// INPUT PINS
//sms signal check pin
const int sigStrength = 1;

// admin control buttons
const int buttonPinAdmin = 2;   // admin functionality enable
const int buttonPinInc = 3;    // button to increment the selection
const int buttonPinAcpt = 4;    // button to move the cursor
const int buttonPinReset = 5;    // button to reset to first position
const int buttonPinMemory = 6;  //button to write to memory
```

```
//temp and humidity module
const int envCheck = 11;  //environment check trigger

//light sensor
const int lightSensor = A7;

//Tone output pin
const int beepPin = 14;

// arm button and state variable
const int armValPin = 20;   // ECHO PIN FOR ARM STATUS
const int armPin = 21;      // SYSTEM ARM TRIGGER

// sms send
const int smsSend_A = 22;    // SMS MESSAGE A TRIGGER
const int smsSend_B = 23;   // SMS MESSAGE B TRIGGER


// OUTPUT PINS
const int ledPin = 24;      // SYSTEM ALARMED LED
const int statusLED = 25;   // SYSTEM CONNECTED TO NETWOK (SIM)
const int smsSimLED_A = 26;   // SMS A TRANSMITTING LED
const int smsSimLED_B = 27;   // SMS B TRANSMITTING LED
const int armLED = 28;      // SYSTEM ARMED LED
const int disarmLED = 29;   // SYSTEM DISARMED LED
const int textRecieve = 30;   // SMS CONTROL OVERRIDE LED

//++++++++++++++++++++

// VARIABLES FOR STATUS
int triggerState_A = 0;         // current state button A
int lastTriggerState_A = 0;     // previous state button A
int triggerState_B = 0;         // current state button B
int lastTriggerState_B = 0;     // previous state button B

// VARIABLE FOR TEXT ARM STATUS
String armState;
int armVal = 0;

//VARIABLE FOR LIGHT VALUES
int photocellValue = 0; //sensor value
int lightBrightness = 0; //calibrated value
String lightState;

// VARIABLE FOR TEXT TEMP
String tempState;

//VARIABLE FOR TEXT HUMIDITY
String humidState;

// VARIABLE TO STORE THE TEXT MESSAGE
String textMessage;

// variable to store SIM900 CSQ response
String recieveCSQ;

//variables for sms number in memory
```

```
int phnAddrA = 0;
int phnAddrB = 1;
int phnAddrC = 2;
int phnAddrD = 3;
int phnAddrE = 4;
int phnAddrF = 5;
int phnAddrG = 6;
int phnAddrH = 7;
int phnAddrI = 8;
int phnAddrJ = 9;

//variable for phone number position
int numPos = 0;

//variables for status POSITION
int State_numPos = 0;
int lastState_numPos = 0;

//variables for reset POSITION
int State_numReset = 0;
int lastState_numReset = 0;

//variables for memory POSITION
int State_Memory = 0;
int lastState_Memory = 0;

//phone number variables
byte A = 0;
byte B = 1;
byte C = 2;
byte D = 3;
byte E = 4;
byte F = 5;
byte G = 6;
byte H = 7;
byte I = 8;
byte J = 9;

//buffer for the phone number
char buffer[50]; //  changed from 10

//variables for status
int State_A = 0;
int lastState_A = 0;

int State_B = 0;
int lastState_B = 0;

int State_C = 0;
int lastState_C = 0;

int State_D = 0;
int lastState_D = 0;

int State_E = 0;
int lastState_E = 0;
```

```
int State_F = 0;
int lastState_F = 0;

int State_G = 0;
int lastState_G = 0;

int State_H = 0;
int lastState_H = 0;

int State_I = 0;
int lastState_I = 0;

int State_J = 0;
int lastState_J = 0;

//++++++++++++++++++++
//TONE FUNCTIONS

//TONE_A
void beepA() {
  tone(beepPin, 554, 250);
  delay (350);
  tone(beepPin, 554, 250);
}
//TONE_B
void beepB() {
  tone(beepPin, 830, 250);
  delay (350);
  tone(beepPin, 830, 250);
}
//TONE_C
void beepC() {
  tone(beepPin, 440, 1000);
}
//++++++++++++++++++++
// SMS FUNCTIONS

// SMS_A_SIMULATED
void smsSimulated_A() {
  //simualte send sms
  digitalWrite(smsSimLED_A, HIGH);
  beepA();
  delay (1000);
  digitalWrite(smsSimLED_A, LOW);
}

//SMS_B_SIMULATED
void smsSimulated_B() {
  //simualte send sms
  digitalWrite(smsSimLED_B, HIGH);
  beepB();
  delay (1000);
  digitalWrite(smsSimLED_B, LOW);
}

//SMS SEND_A
void sendSMS_A() {
```

```
  // AT command to set SIM900 to SMS mode
  SIM900.print("AT+CMGF=1\r");
  delay(100);

  // mobile phone number to send SMS to
sprintf(buffer, "AT + CMGS = \"+1""%d%d%d%d%d%d%d%d%d%d\"", A, B, C, D, E, F,
G, H, I, J); //eeprom
Serial.println(buffer); //eeprom
SIM900.println(buffer);//eeprom
  //SIM900.println("AT + CMGS = \"+18599926861\""); //HARD CODE NUMBER

  delay(100);

  // SMS_A message content
  SIM900.println("This is test message A - sent via a PLP-1.");
  delay(100);

  // End AT command with a ^Z, ASCII code 26
  SIM900.println((char)26);
  delay(100);
  SIM900.println();
  // SMS send timeout
  delay(5000);
}

//SMS SEND_B
void sendSMS_B() {
  // AT command to set SIM900 to SMS mode
  SIM900.print("AT+CMGF=1\r");
  delay(100);

  // mobile phone number to send SMS to
  SIM900.println("AT + CMGS = \"+18599926861\"");
  delay(100);

  // SMS_B message content
  SIM900.println("This is test message B - sent via a PLP-1.");
  delay(100);

  // End AT command with a ^Z, ASCII code 26
  SIM900.println((char)26);
  delay(100);
  SIM900.println();
  // SMS send delay
  delay(5000);
}
//SMS RESPOND HANDLES SMS SEND ONLY NOT CONTENT
//SMS RESPOND CONTROL_A - SYSTEM STATE
void respondSMS_A(String message) {
  // AT command to set SIM900 to SMS mode
  SIM900.print("AT+CMGF=1\r");
  delay(100);

  // mobile phone number to send SMS to
  SIM900.println("AT + CMGS = \"+18599926861\"");
  delay(100);
  // Send the SMS
```

```
  SIM900.println(message);
  delay(100);

  // End AT command with a ^Z, ASCII code 26
  SIM900.println((char)26);
  delay(100);
  SIM900.println();

  // Give module time to send SMS
  delay(5000);
}

//SMS RESPOND CONTROL_B - ENVIRONMENT STATE
void respondSMS_B(String message) {
  // AT command to set SIM900 to SMS mode
  SIM900.print("AT+CMGF=1\r");
  delay(100);

  // mobile phone number to send SMS to
  SIM900.println("AT + CMGS = \"+18599926861\"");
  delay(100);

  // Send the SMS
  SIM900.println(message);
  delay(100);

  // End AT command with a ^Z, ASCII code 26
  SIM900.println((char)26);
  delay(100);
  SIM900.println();

  // Give module time to send SMS
  delay(5000);
}

//THIS IS WHERE THE CONTENT OF THE RESPONSES TO SMS INQUIRY ARE CREATED
// TEXT CONTROL RECEIVE
void textControl() {

  if (SIM900.available() > 0) {
    textMessage = SIM900.readString();
    delay(10);
  }
  if (textMessage.indexOf("ON") >= 0) {
    // Set to ARMED save current state
    digitalWrite(textRecieve, HIGH);
    digitalWrite(armPin, HIGH);
    armState = "ARMED";
    textMessage = "";
  }
  if (textMessage.indexOf("OFF") >= 0) {
    // Set to DISARMED and save current state
    digitalWrite(textRecieve, LOW);
    digitalWrite(armPin, LOW);
    armState = "DISARMED";
    textMessage = "";
```

```
  }

  if (textMessage.indexOf("STATE") >= 0) {

    String message = "System is " + armState;
    respondSMS_A(message);
    textMessage = "";
  }

    if (textMessage.indexOf("ENV") >= 0) {

    // get temperature event
    sensors_event_t event;
    dht.temperature().getEvent(&event);

    //VARIABLE FOR TEMPERATURE
    tempState = (event.temperature * 1.8 + 32); //convert to F
    //tempState = event.temperature;

      // Get humidity event
  dht.humidity().getEvent(&event);

  //variable for humidity
  humidState = event.relative_humidity;

   // read the value from the sensor:
    photocellValue = analogRead(lightSensor);
    photocellValue = constrain(photocellValue, 200, 800); //adjust depending
on environment.

  int lightBrightness = map(photocellValue, 200, 800, 0, 255);
  if (lightBrightness <=10) {
      lightState = "Lights Off";
    }
      else
      {
          lightState = "Lights On";
      }

    String message = "NKU Robotics Lab: System is " + armState+ ". " + "Temp
is " + tempState + " F. Humidity is " + humidState + "%. " + lightState;
    respondSMS_B(message);
    textMessage = "";
  }
}

// NETWORK STRENGTH CHECK
void textNetwork() {

  //ping the SIM900
  SIM900.println("AT + CSQ");
  delay(1000);

  if (SIM900.available() > 0) {
    recieveCSQ = SIM900.readString();
    delay(10);
```

```
  }

  // Clear the display buffer
  display.clearDisplay();

  // text display tests
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.println(recieveCSQ);
  display.display();
}


//++++++++++++++++++++
// DISPLAY FUNCTIONS

void dispArm() {
  // Clear the buffer.
  display.clearDisplay();

  // text display tests
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.println("System:");
  display.println("");
  display.println("ARMED");
  display.display();
}

void dispDisarm() {
  // Clear the buffer.
  display.clearDisplay();

  // text display tests
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.println("System:");
  display.println("");
  display.println("DISARMED");
  display.display();
}

void dispLogo() {
  // Clear the buffer.
  display.clearDisplay();

  // text display tests
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.println("");
  display.println("     Created By:");
  display.println("");
  display.setTextSize(2);
```

```
  display.setTextColor(WHITE);
  display.println(" Encom Lab");

  display.display();
}

void dispAlarmA() {
  // Clear the buffer.
  display.clearDisplay();

  // text display tests
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.println("");
  display.println("ALARM A");
  display.println("");
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.println("MESSAGE TX");

  display.display();
}

void dispAlarmB() {
  // Clear the buffer.
  display.clearDisplay();

  // text display tests
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.println("");
  display.println("ALARM B");
  display.println("");
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.println("MESSAGE TX");

  display.display();
}

void dispPhoneNumber() {
  // Clear the display buffer
  display.clearDisplay();
  //oled
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.println("Phone Number A:");
  display.setCursor(0, 20);

  //formatted number
  display.print("#: ");
  display.print(A);
  display.print(B);
  display.print(C);
```

```
  display.print("-");
  display.print(D);
  display.print(E);
  display.print(F);
  display.print("-");
  display.print(G);
  display.print(H);
  display.print(I);
  display.print(J);

  switch (numPos) {
    case 0:
      numHighlightA();
      break;
    case 1:
      numHighlightB();
      break;
    case 2:
      numHighlightC();
      break;
    case 3:
      numHighlightD();
      break;
    case 4:
      numHighlightE();
      break;
    case 5:
      numHighlightF();
      break;
    case 6:
      numHighlightG();
      break;
    case 7:
      numHighlightH();
      break;
    case 8:
      numHighlightI();
      break;
    case 9:
      numHighlightJ();
      break;
  }
  display.display();
}

void dispSaveA() {
  // Clear the buffer.
  display.clearDisplay();

  // text display tests
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.println("Number A");
```

```
  display.println("");
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.println("SAVED");
  display.display();
  delay(2000);
}

//RESET BUTTON
void resetPos() {
  State_numReset = digitalRead(buttonPinReset);
  // compare the buttonState to its previous state
  if (State_numReset != lastState_numReset) {
    // if the state has changed, increment the counter
    if (State_numReset == HIGH) {
      // if the current state is HIGH then the button went from off to on:
      //digitalWrite(ledPin, HIGH);
      //reset the number position to 0
      numPos = 0;
    } else {
      // if the current state is LOW then the button went from on to off:
      //digitalWrite(ledPin, LOW);
    }
    //debounce delay
    delay(50);
  }
  // save the current state as the last state, for next time through the loop
  lastState_numReset = State_numReset;
}

//READ INPUT PIN POSITION
void inputPos() {
  State_numPos = digitalRead(buttonPinAcpt);
  // compare the buttonState to its previous state
  if (State_numPos != lastState_numPos) {
    // if the state has changed, increment the counter
    if (State_numPos == HIGH) {
      // if the current state is HIGH then the button went from off to on:
      //digitalWrite(ledPin, HIGH);
      //increment the number
      numPos++;
    } else {
      // if the current state is LOW then the button went from on to off:
      //digitalWrite(ledPin, LOW);
    }
    //debounce delay
    delay(50);
  }
  // save the current state as the last state, for next time through the loop
  lastState_numPos = State_numPos;

  //reset the variable after 9
  if (numPos == 10) {
    numPos = 0;
  }
}
```

```
//READ INPUT PIN MEMORY
void inputMemory() {
  State_Memory = digitalRead(buttonPinMemory);
  // compare the buttonState to its previous state
  if (State_Memory != lastState_Memory) {
    // if the state has changed, increment the counter
    if (State_Memory == HIGH) {
      // if the current state is HIGH then the button went from off to on:
      //digitalWrite(ledPinAdmin, HIGH);

      phnEepromWrite();
      dispSaveA();

      //J++;
    } else {
      // if the current state is LOW then the button went from on to off:
      //digitalWrite(ledPinAdmin, LOW);
    }
    // debounce delay
    delay(50);
  }
  // save the current state as the last state, for next time through the loop
  lastState_Memory = State_Memory;
}


//READ INPUT PIN A
void inputA() {

  State_A = digitalRead(buttonPinInc);
  // compare the buttonState to its previous state
  if (State_A != lastState_A) {
    // if the state has changed, increment the counter
    if (State_A == HIGH) {
      // if the current state is HIGH then the button went from off to on:
      //digitalWrite(ledPin, HIGH);
      //increment the number
      A++;
    } else {
      // if the current state is LOW then the button went from on to off:
      // digitalWrite(ledPin, LOW);
    }
    // debounce delay
    delay(50);
  }
  // save the current state as the last state, for next time through the loop
  lastState_A = State_A;
  //reset the variable after 9
  if (A == 10) {
    A = 0;
  }
}

//READ INPUT PIN B
void inputB() {
```

```
  State_B = digitalRead(buttonPinInc);
  // compare the buttonState to its previous state
  if (State_B != lastState_B) {
    // if the state has changed, increment the counter
    if (State_B == HIGH) {
      // if the current state is HIGH then the button went from off to on:
      //digitalWrite(ledPin, HIGH);
      B++;
    } else {
      // if the current state is LOW then the button went from on to off:
      //digitalWrite(ledPin, LOW);
    }
    // debounce delay
    delay(50);
  }
  // save the current state as the last state, for next time through the loop
  lastState_B = State_B;
  //reset the variable after 9
  if (B == 10) {
    B = 0;
  }
}

//READ INPUT PIN C
void inputC() {
  State_C = digitalRead(buttonPinInc);
  // compare the buttonState to its previous state
  if (State_C != lastState_C) {
    // if the state has changed, increment the counter
    if (State_C == HIGH) {
      // if the current state is HIGH then the button went from off to on:
      //digitalWrite(ledPin, HIGH);
      C++;
    } else {
      // if the current state is LOW then the button went from on to off:
      // digitalWrite(ledPin, LOW);
    }
    // debounce delay
    delay(50);
  }
  // save the current state as the last state, for next time through the loop
  lastState_C = State_C;
  //reset the variable after 9
  if (C == 10) {
    C = 0;
  }
}

//READ INPUT PIN D
void inputD() {
  State_D = digitalRead(buttonPinInc);
  // compare the buttonState to its previous state
  if (State_D != lastState_D) {
    // if the state has changed, increment the counter
    if (State_D == HIGH) {
      // if the current state is HIGH then the button went from off to on:
```

```
      //digitalWrite(ledPin, HIGH);
      D++;
    } else {
      // if the current state is LOW then the button went from on to off:
      // digitalWrite(ledPin, LOW);
    }
    // debounce delay
    delay(50);
  }
  // save the current state as the last state, for next time through the loop
  lastState_D = State_D;
  //reset the variable after 9
  if (D == 10) {
    D = 0;
  }
}

//READ INPUT PIN E
void inputE() {
  State_E = digitalRead(buttonPinInc);
  // compare the buttonState to its previous state
  if (State_E != lastState_E) {
    // if the state has changed, increment the counter
    if (State_E == HIGH) {
      // if the current state is HIGH then the button went from off to on:
      //digitalWrite(ledPin, HIGH);
      E++;
    } else {
      // if the current state is LOW then the button went from on to off:
      // digitalWrite(ledPin, LOW);
    }
    // debounce delay
    delay(50);
  }
  // save the current state as the last state, for next time through the loop
  lastState_E = State_E;
  //reset the variable after 9
  if (E == 10) {
    E = 0;
  }
}

//READ INPUT PIN F
void inputF() {
  State_F = digitalRead(buttonPinInc);
  // compare the buttonState to its previous state
  if (State_F != lastState_F) {
    // if the state has changed, increment the counter
    if (State_F == HIGH) {
      // if the current state is HIGH then the button went from off to on:
      // digitalWrite(ledPin, HIGH);
      F++;
    } else {
      // if the current state is LOW then the button went from on to off:
      // digitalWrite(ledPin, LOW);
    }
```

```
    // debounce delay
    delay(50);
  }
  // save the current state as the last state, for next time through the loop
  lastState_F = State_F;
  //reset the variable after 9
  if (F == 10) {
    F = 0;
  }
}

//READ INPUT PIN G
void inputG() {
  State_G = digitalRead(buttonPinInc);
  // compare the buttonState to its previous state
  if (State_G != lastState_G) {
    // if the state has changed, increment the counter
    if (State_G == HIGH) {
      // if the current state is HIGH then the button went from off to on:
      //digitalWrite(ledPin, HIGH);
      G++;
    } else {
      // if the current state is LOW then the button went from on to off:
      // digitalWrite(ledPin, LOW);
    }
    // debounce delay
    delay(50);
  }
  // save the current state as the last state, for next time through the loop
  lastState_G = State_G;
  //reset the variable after 9
  if (G == 10) {
    G = 0;
  }
}

//READ INPUT PIN H
void inputH() {
  State_H = digitalRead(buttonPinInc);
  // compare the buttonState to its previous state
  if (State_H != lastState_H) {
    // if the state has changed, increment the counter
    if (State_H == HIGH) {
      // if the current state is HIGH then the button went from off to on:
      // digitalWrite(ledPin, HIGH);
      H++;
    } else {
      // if the current state is LOW then the button went from on to off:
      // digitalWrite(ledPin, LOW);
    }
    // debounce delay
    delay(50);
  }
  // save the current state as the last state, for next time through the loop
  lastState_H = State_H;
  //reset the variable after 9
```

```
  if (H == 10) {
    H = 0;
  }
}

//READ INPUT PIN I
void inputI() {
  State_I = digitalRead(buttonPinInc);
  // compare the buttonState to its previous state
  if (State_I != lastState_I) {
    // if the state has changed, increment the counter
    if (State_I == HIGH) {
      // if the current state is HIGH then the button went from off to on:
      // digitalWrite(ledPin, HIGH);
      I++;
    } else {
      // if the current state is LOW then the button went from on to off:
      //  digitalWrite(ledPin, LOW);
    }
    // debounce delay
    delay(50);
  }
  // save the current state as the last state, for next time through the loop
  lastState_I = State_I;
  //reset the variable after 9
  if (I == 10) {
    I = 0;
  }
}
//READ INPUT PIN J
void inputJ() {
  State_J = digitalRead(buttonPinInc);
  // compare the buttonState to its previous state
  if (State_J != lastState_J) {
    // if the state has changed, increment the counter
    if (State_J == HIGH) {
      // if the current state is HIGH then the button went from off to on:
      //  digitalWrite(ledPin, HIGH);
      J++;
    } else {
      // if the current state is LOW then the button went from on to off:
      //  digitalWrite(ledPin, LOW);
    }
    // debounce delay
    delay(50);
  }
  // save the current state as the last state, for next time through the loop
  lastState_J = State_J;
  //reset the variable after 9
  if (J == 10) {
    J = 0;
  }
}

//CURSOR FUNCTIONS
void numHighlightA() {
```

```
  display.setCursor(18, 22);
  display.print("_");
}
void numHighlightB() {
  display.setCursor(24, 22);
  display.print("_");
}
void numHighlightC() {
  display.setCursor(30, 22);
  display.print("_");
}
void numHighlightD() {
  display.setCursor(42, 22);
  display.print("_");
}
void numHighlightE() {
  display.setCursor(48, 22);
  display.print("_");
}
void numHighlightF() {
  display.setCursor(54, 22);
  display.print("_");
}
void numHighlightG() {
  display.setCursor(66, 22);
  display.print("_");
}
void numHighlightH() {
  display.setCursor(72, 22);
  display.print("_");
}
void numHighlightI() {
  display.setCursor(78, 22);
  display.print("_");
}
void numHighlightJ() {
  display.setCursor(84, 22);
  display.print("_");
}

void smsPhoneNumA() { //EEPROM TEST
  sprintf(buffer, "+1""%d%d%d%d%d%d%d%d%d%d", A, B, C, D, E, F, G, H, I, J);
//EEPROM TEST
  Serial.println(buffer); //EEPROM TEST
}

void numPosition() {
  //switch case to move through phone number
  switch (numPos) {
    case 0:
      inputA();
      break;
    case 1:
      inputB();
      break;
    case 2:
```

```
      inputC();
      break;
    case 3:
      inputD();
      break;
    case 4:
      inputE();
      break;
    case 5:
      inputF();
      break;
    case 6:
      inputG();
      break;
    case 7:
      inputH();
      break;
    case 8:
      inputI();
      break;
    case 9:
      inputJ();
      break;
  }
}

//EEPROM WRITE PHONE NUMBER
void phnEepromWrite() {
  //memory write to phnAddress with value of A
  //short delay is for write time
  EEPROM.put(phnAddrA, A);
  delay(50);
  EEPROM.put(phnAddrB, B);
  delay(50);
  EEPROM.put(phnAddrC, C);
  delay(50);
  EEPROM.put(phnAddrD, D);
  delay(50);
  EEPROM.put(phnAddrE, E);
  delay(50);
  EEPROM.put(phnAddrF, F);
  delay(50);
  EEPROM.put(phnAddrG, G);
  delay(100);
  EEPROM.put(phnAddrH, H);
  delay(100);
  EEPROM.put(phnAddrI, I);
  delay(100);
  EEPROM.put(phnAddrJ, J);
  delay(100);
}

//EEPROM READ PHONE NUMBER
void phnEepromRead() {
```

```
  A = EEPROM.get(phnAddrA, A);
  B = EEPROM.get(phnAddrB, B);
  C = EEPROM.get(phnAddrC, C);
  D = EEPROM.get(phnAddrD, D);
  E = EEPROM.get(phnAddrE, E);
  F = EEPROM.get(phnAddrF, F);
  G = EEPROM.get(phnAddrG, G);
  H = EEPROM.get(phnAddrH, H);
  I = EEPROM.get(phnAddrI, I);
  J = EEPROM.get(phnAddrJ, J);

}

//TEMP AND HUMIDITY SENSOR
void tempHumidityCheck() {

  // clear the display buffer
  display.clearDisplay();

  // set oled font
  display.setTextSize(2);
  display.setTextColor(WHITE);

  // get temperature event and print its value
  sensors_event_t event;
  dht.temperature().getEvent(&event);

  // display temperature
  display.setCursor(0, 0);
  display.println("Temp:");
  display.println("");
  display.print(event.temperature * 1.8 + 32); //convert to F
  display.print(" *F");
  display.display();
  //}
  // Get humidity event and print its value.
  dht.humidity().getEvent(&event);

  delay (3000);
  // Clear the display buffer
  display.clearDisplay();
  display.setCursor(0, 0);
  display.println("Humidity: ");
  display.println("");
  display.print(event.relative_humidity);
  display.print("%");
  display.display();
  // }
  delay (3000);
  // Clear the display buffer
  display.clearDisplay();
}

//LIGHT SENSOR CHECK
void lightCheck(){
     // read the value from the sensor:
    photocellValue = analogRead(lightSensor);
```

```
    photocellValue = constrain(photocellValue, 200, 800); //adjust depending
on environment.

  int lightBrightness = map(photocellValue, 200, 800, 0, 255);
  if (lightBrightness <=10) {
      lightState = "Lights Off";
  }
      else
      {
          lightState = "Lights On";
      }

  // Clear the display buffer
  display.clearDisplay();

  // text display tests
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.println(lightState);
  display.display();

}

//TIME CHECK
void timeCheck(){

     //ping the SIM900
    SIM900.println("AT+CCLK?");
    delay(300);

  if (SIM900.available() > 0) {
    recieveCSQ = SIM900.readString();
    delay(10);
  }

  // Clear the display buffer
  display.clearDisplay();

  // text display tests
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.println(recieveCSQ);
  display.display();
}

void disableWatchdog()
{
  cli();
  wdt_reset();
  MCUSR &= ~(1<<WDRF);
  WDTCSR |= (1<<WDCE) | (1<<WDE);
  WDTCSR = 0x00;
  sei();
}
```

```
//++++++++++++++++++++

//++++++++++++++++++++

// SETUP START +++++
void setup() {
  //this function clears the eeprom
  //  for (int i = 0 ; i < EEPROM.length() ; i++) {
  //   EEPROM.write(i, 0);
  // }

//disable the wdt
  disableWatchdog();

  // INPUT SETUP
  // initialize the button pin as a input:
  pinMode(smsSend_A, INPUT);
  pinMode(smsSend_B, INPUT);
  pinMode(armValPin, INPUT);
  pinMode(sigStrength, INPUT);
  pinMode(envCheck, INPUT);

  pinMode(buttonPinInc, INPUT);
  pinMode(buttonPinAcpt, INPUT);
  pinMode(buttonPinReset, INPUT);
  pinMode(buttonPinAdmin, INPUT);
  pinMode(buttonPinMemory, INPUT);

  //sensor input
  pinMode(DHTPIN, INPUT);
  pinMode(lightSensor, INPUT);

  // OUTPUT SETUP
  // initialize the LED as an output:
  pinMode(ledPin, OUTPUT);
  pinMode(statusLED, OUTPUT);
  pinMode(smsSimLED_A, OUTPUT);
  pinMode(smsSimLED_B, OUTPUT);
  pinMode(armLED, OUTPUT);
  pinMode(disarmLED, OUTPUT);
  pinMode(textRecieve, OUTPUT);
  pinMode(beepPin, OUTPUT);

  // initialize oled power
  display.begin(SH1106_SWITCHCAPVCC, 0x3C);

  // empty display buffer
  display.clearDisplay();

  //display logo
  dispLogo();

  //write memory to phone variables
  phnEepromRead();

  //initialize temp and humidity sensor
```

```
  dht.begin();

  // sim 900 initialization
  SIM900.begin(19200);

  // DELAY FOR SIM900 NETWORK STARTUP
  delay(20000);

  // CONNECTION STATUS LED
  digitalWrite(statusLED, HIGH);
  //PLAY START TONE
  beepC();

  //serial for testing
  //Serial.begin(19200);
  //Serial.println("SIM900 ready...");

  // AT command to set SIM900 to SMS mode
  SIM900.print("AT+CMGF=1\r");
  delay(100);

  // Set module to send SMS data to serial out upon receipt
  SIM900.print("AT+CNMI=2,2,0,0,0\r");
  delay(100);
}
// SETUP END -----

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// MAIN LOOP START +++++
void loop() {

//obtain system state values
  armVal = digitalRead(armValPin);
  if (armVal == HIGH) {
    armState = "ARMED";
  }
  else if (armVal == LOW) {
    armState = "DISARMED";
  }

  // text control function
  textControl();

  // check signal strenght
  while (digitalRead(sigStrength) == HIGH) {
    textNetwork();
  }

  //check environment status
  while (digitalRead(envCheck) == HIGH) {
    tempHumidityCheck();
  }

  // system arm status
  while (digitalRead(armPin) == HIGH) {
```

```
    armVal = digitalRead(armValPin);
    if (armVal == HIGH) {
      armState = "ARMED";
    }
    else if (armVal == LOW) {
      armState = "DISARMED";
    }

    // text control function
    textControl();

    // arm status led
    digitalWrite(armLED, HIGH);
    digitalWrite(disarmLED, LOW);
    //arm status oled
    dispArm();

    //READ INPUT PIN A
    triggerState_A = digitalRead(smsSend_A);
    // compare the buttonState to its previous state
    if (triggerState_A != lastTriggerState_A) {
      // if the state has changed, increment the counter
      if (triggerState_A == HIGH) {
        // if the current state is HIGH then the button went from off to on:
        digitalWrite(ledPin, HIGH);

        //simualte send sms
        smsSimulated_A();

        // Send the SMS
        sendSMS_A();
        //DISPLAY ALARM STATUS A
        dispAlarmA();

      } else {
        // if the current state is LOW then the button went from on to off:
        digitalWrite(ledPin, LOW);
      }
      // Delay a little bit to avoid bouncing
      delay(50);
    }
    // save the current state as the last state, for next time through the
loop
    lastTriggerState_A = triggerState_A;
    //END STATE CHECK A

    //READ INPUT PIN B
    triggerState_B = digitalRead(smsSend_B);

    // compare the buttonState to its previous state
    if (triggerState_B != lastTriggerState_B) {
      // if the state has changed, increment the counter
      if (triggerState_B == HIGH) {
        // if the current state is HIGH then the button went from off to on:
        digitalWrite(ledPin, HIGH);
```

```
        //simualte send sms
        smsSimulated_B();

        // SEND THE SMS
        sendSMS_B();
        //DISPLAY ALARM STATUS
        dispAlarmB();

      } else {
        // if the current state is LOW then the button went from on to off:
        digitalWrite(ledPin, LOW);
      }
      // Delay a little bit to avoid bouncing
      delay(50);
    }
    // save the current state as the last state, for next time through the
loop
    lastTriggerState_B = triggerState_B;
  }

  // arm status led
  digitalWrite(armLED, LOW);
  digitalWrite(disarmLED, HIGH);
  //oled display disarm
  dispDisarm();

  //system admin status
  while (digitalRead(buttonPinAdmin) == HIGH) {

    //function to determine phone A number position
    inputPos();

    //function to reset phone A position
    resetPos();

    //function to set memory write
    inputMemory();

    //function to select the input for the position
    numPosition();

    //call display function
    dispPhoneNumber();
  }
}
// MAIN LOOP END -----
```

References

[1] "OSHA Technical Manual (OTM) | Section IV: Chapter 4 ...," [Online]. Available: https://www.osha.gov/dts/osta/otm/otm_iv/otm_iv_4.html#app_iv:4_1. [Accessed 16 5 2018].

[2] SRI International, "SRI International," 12 April 2018. [Online]. Available: https://www.sri.com/work/timeline-innovation/timeline.php?timeline=computing-digital#!&innovation=shakey-the-robot.

[3] I. K. Olarewaju, O. E. Ayodele, F. O. Michael, E. S. Alaba and R. O. Abiodun, "Design and Construction of an Automatic Home Security System Based on GSM Technology and Embedded Microcontroller Unit," 2017. [Online]. Available: http://article.sciencepublishinggroup.com/pdf/10.11648.j.ece.20170101.14.pdf. [Accessed 5 8 2018].

[4] X. X. Zheng, L. R. Li and Y. J. Shao, "A GSM-Based Remote Temperature and Humidity Monitoring System for Granary," 2016. [Online]. Available: https://matec-conferences.org/articles/matecconf/ref/2016/07/matecconf_iceice2016_01060/matecconf_iceice2016_01060.html. [Accessed 16 5 2018].