EE 3420 – Microprocessors Laboratory Manual

Fall 2022 Edition

Revised 07/08/2022 – W. A. Stapleton

Texas State University

Ingram School of Engineering

San Marcos, Texas

General Laboratory Information

Equipment:
The laboratory assignments for EE 3420 will require access to a microprocessor board and peripheralss. Prior to the beginning of the laboratory, you will need to purchase your microprocessor board with appropriate accessories.

The recommended laboratory kit consists of the "REXQualis Mega 2560 Kit: The Most Complete Starter Kit" which is most easily purchased through Amazon. You can purchase any kit which contains the needed components. The lectures and labs for the class will utilize as much of the contents of this kit as possible given the time available in the class. Perhaps of equal importance is that these items were selected so that they would likely be useful in other classes, especially in EE 4390 and EE 4391, the Senior Design sequence. The trends over the last few years have shown that the majority of the design projects need microprocessors.

Schedule:
The laboratory will meet in sections. The initial section schedule is available in the Schedule of Classes booklet. This schedule may need to be modified in response to the evolving Coronavirus situation. We will make every effort to follow the best practices identified by the University for safeguarding everyone's health.

One of the purposes of having each student purchase the full parts kit is to allow for the possibility of students working remotely to avoid close quarters with others in a laboratory room.

The intent of your laboratory meeting time is for it to be the time set aside ***for demonstration of your completed projects***. There is almost never sufficient time in a three-hour session to start and complete a project. Another purpose of having students purchase the laboratory kits is to allow students to work on the projects at their convenience throughout the week without requiring them to come into the classroom.

Please note that you must demonstrate your laboratory assignment during (or before) the lab session on the due date in order to receive credit for the laboratory.

Please be prepared to demonstrate your project at your scheduled time since the available diagnostic equipment is limited and the number of students who can be accommodated at any given time is limited. More detailed information on the due dates for each specific assignment is available on the class website.

Website:
The class website is available via the Canvas system. You should be automatically enrolled.

Individual vs. Group Work:
One of the outcomes required for ABET accreditation is assessing a student's ability to work in a group setting. The projects in this class are a mixture of individual and group work. Each laboratory will indicate its appropriate group size. The early laboratories will be designed as

individual projects. Every student will complete the project separately and turn in a separate laboratory report. The later laboratories will require a degree of collaboration, but this will be managed such that the projects will be built as two or more interworking parts. The students working in a group will need to communicate with each other in order to make the various parts of the project work together.

For the group laboratories, duties as group leader **must** be rotated among all members of the group. No one should serve as group leader on two subsequent labs. Every person must serve as group leader on at least one project. The primary duties for the group leader are to coordinate the activities of the other group members, distribute the workload equitably among all members of the group, and bear ultimate responsibility for ensuring that the project is completed on time and that the report is complete and properly formatted. The group leader is also responsible for completing a one-page summary detailing the contribution from each group member to the project. A sample of this summary page is included and may be completed by hand rather than typed. This summary does not need to be included for individual projects. This summary page will be appended to the report.

Should the need for strong social distancing measures to prevent Coronavirus transmission become apparent, project collaboration will need to be modified. The labs are capable of being modified so that every individual will work separately but that they may "collaborate" by swapping code with one another and making certain that the programs still work. If both members of the "group" followed the communication protocol correctly, they should be able to swap code components seamlessly.

Report Preparation:

After the successful conclusion of each laboratory exercise, a laboratory report will be due. The typical due date for the report will be at the beginning of the next laboratory period.

A laboratory report must be submitted in order to receive any credit for the lab.

There will be two types of laboratory reports. For the early, simpler labs an informal report will be sufficient. However, as EE 3420 is designated as "Writing Intensive", several reports for the more complex labs, specifically those requiring teams, will be formal in nature. The class website includes a document describing the laboratory report format in detail. This document may also be used as a template for your report.

Code Documentation:

Code is only correct if it both does its intended task correctly **and** it is properly documented. Proper documentation contains, **at minimum**, the following elements:

- In the main program:
  - The name(s) of the author(s) of any portion of the program
  - The name of the program (*e.g.* lab1.c)
  - A brief description of the function of the program

    - A notation of any special considerations which need to be made for compiling or running the program, *e.g.*, included libraries and/or compiler arguments, connected hardware, etc.
    - A listing of all peripheral devices needed and a listing of all of the port pin connections which need to be made. (*e.g.* microprocessor pin 17 to the red LED anode)
    - A listing of any external libraries or other files needed to compile the program.
- In each subroutine, function, or ISR:
    - The name(s) of the author(s) of the routine.
    - A brief description of the function of the routine.
    - A complete listing of the size, number, and type of parameters passed as input to the function and generated as output from the function.
    - Any hardware-specific information not included with the main program.
- For any code:
    - In-line comments describing what each portion of the code does.
    - These comments would likely correspond to the blocks and decision points on a flowchart of the program.

Please remember that code which is not properly documented will not be considered correct or complete and the assignment will receive a grade of <u>zero</u>.

Laboratory Grading:

The grade for each laboratory assignment will consist of four major factors: a quiz on pre-lab preparation, code functionality, code documentation, and laboratory report. The grade for each assignment will be computed as follows:

In order to receive a grade you must do all of the following …

1. Demonstrate your code
2. Document your code
3. Submit a laboratory report by the due date (5% per day penalty for late reports)

If those three prerequisites are met, then the lab grade is calculated as follows …

70% Project functionality

- This portion of the grade will be used to assess whether each assigned task for the laboratory has been accomplished and will include both assessments of the correctness of software code and hardware configurations. Credit for the laboratory may not be given without a code demonstration.

10% Code documentation

- This portion of the grade will be used to assess the documentation of the code written for the laboratory assignment. If the minimum requirements for code documentation listed earlier in this manual are not met, the code will not be considered correct and the assignment will receive a grade of zero.

20% Laboratory report

- The laboratory report must follow the format specified in the report format document on the class website. The narrative content of the report must include a full description of the programs written including descriptions of all subroutines as well as a full description of all hardware used including wiring diagrams. The group leader's summary page and the code listing signed by the laboratory instructor must be appended to the report. If the laboratory report fails to follow the template or the required appendices are missing, the laboratory assignment will receive a grade of zero.

Summary of Contributions

Laboratory # _____

Group Leader (Team Member #1)

Name: _____ Student Number: _____

Please list all contributions to the project (*e.g.* main program, subroutines by name, etc.)

_____
_____
_____
_____
_____
_____

Team Member #2

Name: _____ Student Number: _____

Please list all contributions to the project (*e.g.* main program, subroutines by name, etc.)

_____
_____
_____
_____
_____
_____

Team Member #3

Name: _____ Student Number: _____

Please list all contributions to the project (*e.g.* main program, subroutines by name, etc.)

_____
_____
_____
_____
_____
_____

Team Member #4

Name: _____ Student Number: _____

Please list all contributions to the project (*e.g.* main program, subroutines by name, etc.)

_____
_____
_____
_____
_____
_____

EE 3420 Laboratory Assignments

Table of Contents

Laboratory 1
UART Communication
Perfect Triangle Generator

This is an individual assignment.

Background and Related Materials

This assignment will give you practice in programming the Arduino Mega 2560 (Mega) to create an interactive "console" interface through the virtual serial port.

Pre-laboratory Tasks

1. Review the class notes assigned to date. Be certain you are comfortable with all concepts discussed.
2. Perform the example assignments in the class notes.

Laboratory Assignment – General Description

In this laboratory you will be using an Arduino Mega 2560 (Mega) to create an interactive console via the UART. The application implemented will be a calculator to find "perfect" triangles. A "perfect" triangle is a right triangle where all three sides have integer lengths.

Task #1 Instructions

1. You are to write a program which will prompt the user for an integer between 1-65535, i.e. a positive 16-bit integer. This will be assumed to be one of the two legs of a right triangle.
2. Your program will then test all possible 16-bit positive integers as the other leg of the right triangle.
3. If the length of the resultant hypotenuse is also an integer, the resulting triangle is known as a "perfect" triangle. A classic example of this is the 3, 4, 5 triangle.
4. The lengths of the three sides of every perfect triangle with the first leg of the specified length should be displayed on the console, one set per line.

Laboratory Report

1. Please follow the laboratory report guidelines found earlier in this manual.

Laboratory 2
General-Purpose Input-Output (GPIO)
Keypad and Character LCD

This is an assignment for a group of 2.

Laboratory Assignment – General Description

In this laboratory you will be extending the user input and output options using a 4x4 telephone-style keypad and character LCD display. You will also be extending communication using a serial connection to another microprocessor.

You will connect the Arduino Mega 2560 to a host computer using the USB-Serial bridge, to another microprocessor using the Serial1 port, and to both a 4x4 keypad and character LCD connected locally.

Task #1 Instructions

1. The program that you write should look for characters entered through …
    - the 4x4 telephone-style keypad which is connected to the Mega.
    - The USB-serial bridge seen as "Serial"
    - The UART port seen as "Serial1"
2. When a key received from any of these sources, the corresponding character should be displayed both on the local "Serial" console and the character LCD.
3. When a key is received from either "Serial" or the 4x4 keypad, it should be echoed to "Serial1".
4. The program should continue to process input until the '*'/'E' key is pressed three times in sequence with no other keys pressed in-between. At that point, the program should shutdown cleanly.

Laboratory Report

Please follow the laboratory report guidelines found earlier in this manual.

Laboratory 3
Stepper Motor Control

This is an individual assignment.

Background and Related Material

Microprocessor-based Embedded Systems are often used to monitor and control mechanical systems. The primary device for converting electrical energy to mechanical energy is the motor. Embedded systems are capable of controlling a wide variety of different motor types.

All of the motors we will consider operate by routing electrical current through coils of wire to produce magnetic fields which cause a shaft to rotate. Thus, electrical energy is converted to mechanical energy in the form of rotational movement. This rotational movement may be converted to linear movement if necessary. When considering the control of a motor, we will be concerned with several factors. The first is the rotational speed of the motor, expressed in units of revolutions per unit time (e.g. revolutions per second). The second is the position of the rotating shaft, expressed in units of angular displacement from some reference point (e.g. 45º clockwise from the mark). The third is the torque or rotational work produced by the motor, expressed in terms of force times distance (e.g. g*cm or oz*in).

In this laboratory we will explore use of stepper motors. Specifically, we will use a four-phase unipolar stepper motor. Stepper motors are optimized for control of position and, as a result, are generally slower and/or have lower torque than similarly sized DC brushed motors or servo motors.

Pre-laboratory Tasks

1. Review the class notes and online datasheets regarding stepper motors.

Laboratory Assignment – General Description

The overall goal of this assignment is for you to produce a system capable of controlling a four-phase unipolar stepper motor. The system will also use the other interfaces developed in the previous labs.

The Rexqualis Kit includes a stepper motor and a motor drive board using either a ULN2803A or ULN2003A driver chip. For our purposes we can treat either model identically. The GPIO pins of the microprocessor can be used to drive the ULN2003A/ULN2803A input pins which, in turn, will drive the stepper motor coils. The motor and driver are pictured below. This motor included in the kit is designed for use with 5V.

Task #1 Instructions

1. The program that you write is to manage user inputs and to control the stepper motor via the ULN2003A/ULN2803A driver.
2. The program written should set up GPIO interface to the stepper motor.
    - For purposes of testing, when the program begins, the stepper should …
    - Be set to clockwise (i.e. forward through the list of steps)
    - Use single-coil full steps
    - Be set 10 steps per second (100 ms delay between steps)
    - Start stepping
3. A message displaying mode (1-3), direction (+ or -) and number of steps per second should be sent to the serial console and updated whenever these parameters change.
4. When a key has been pressed on either the 4x4 keypad or serial console …
    - If the key is “1”, set the motor to use single-coil full steps (mode 1)
    - If the key is “2”, set the motor to use dual-coil full steps (mode 2)
    - If the key is “3”, set the motor to use half steps, i.e. alternate single-coil and dual-coil steps (mode 3)
    - If the key is “A”, set the direction as clockwise/forward through the list (+ direction)
    - If the key is “B”, set the direction as counter-clockwise/backward through the list (- direction)
    - If the key is “C”, decrease the number of steps per second, minimum is 1 step per second
    - If the key is “D”, increase the number of steps per second, maximum is 1000 steps per second.
    - If the key is ‘*’/“E”, stop taking steps
    - If the key is ‘#’/“F”, start taking steps
    - Any other key should be ignored
    - Whenever the motor status changes due to a key, the message on the console should be updated accordingly
5. The program should continue to process input until the ‘*’/‘E’ key is pressed three times in sequence with no other keys pressed in-between. At that point, the program should shutdown cleanly and cut all power to the motor.

Report Preparation

Prepare your report according to the guidelines presented earlier in the laboratory manual and submit it by the due date.

Laboratory 4
General-Purpose Input-Output (GPIO)
Traffic Control

This is an assignment for a group of 2.

Laboratory Assignment – General Description

In this laboratory you will be using two microprocessors cooperating to create a simulation of a traffic light controller for an intersection. This will consist of a standard Green-Yellow-Red indicator for a North-South road and a similar Green-Yellow-Red indicator for an intersecting East-West road. The situation is complicated somewhat by an "emergency services override" which will stop traffic in both N-S and E-W directions to allow emergency vehicles through. While in this override mode, a Blue indicator will show in each direction. Further, a buzzer will be used as an audible indicator of this mode.

The traffic light system will be accomplished using two microprocessors. One microprocessor will control the North-South lights and the other will control the East-West lights. The microprocessor chosen to control the N-S lights will also determine the state changes. The microprocessor chose to control the E-W lights will follow the lead of the first microprocessor. Each microprocessor will have its own connection to a host computer, a connection to the other via Serial1, and its own keypad and LCD.

The Traffic Light State Machine

The traffic light system should operate in the manner of a state machine.
States:

1. Both N-S Red and E-W Red are lighted. All other LED off. Remain in this state 3 seconds. If the button has been pressed since the last check, activate the override mode. If override is not active, go to state 2. If override is active, go to state 9.
2. Both N-S Green and E-W Red are lighted. All other LED off. If the button has been pressed since the last check, activate the override mode. Remain in this state 7 seconds if override is not active or 2 seconds if override is active. If override is not active, go to state 3. If override is active, go to state 7.
3. Both N-S Yellow and E-W Red are lighted. All other LED off. If the button has been pressed since the last check, activate the override mode. Remain in this state 2 seconds. If override is not active, go to state 4. If override is active, go to state 9.
4. Both N-S Red and E-W Red are lighted. All other LED off. Remain in this state 3 seconds. If the button has been pressed since the last check, activate the override mode. If override is not active, go to state 5. If override is active, go to state 9.
5. Both N-S Red and E-W Green are lighted. All other LED off. If the button has been pressed since the last check, activate the override mode. Remain in this state 7 seconds if override is not active or 2 seconds if override is active. If override is not active, go to state 6. If override is active, go to state 8.

6. Both N-S Red and E-W Yellow are lighted. All other LED off. If the button has been pressed since the last check, activate the override mode. Remain in this state 2 seconds. If override is not active, go to state 1. If override is active, go to state 9.
7. Both N-S Yellow and E-W Red are lighted. All other LED off. Remain in this state 2 seconds. Go to state 9.
8. Both N-S Red and E-W Yellow are lighted. All other LED off. Remain in this state 2 seconds. Go to state 9.
9. Both N-S Red and E-W Red are lighted. Both N-S Blue and E-W Blue are lighted. All other LED off. Turn the buzzer on for 1 second then turn it off for 1 second. Repeat the buzzer cycle for a total of five times for a total of 10 seconds. Before leaving this state, force override mode to be inactive and be certain to turn the buzzer off. Go to state 1.

Task #1 Instructions

1. The program that you write for the first microprocessor will serve to directly control the North-South lights and determine the state progression.
2. The program that you write should begin with placing the the traffic light system in state 1 and immediately start into the specified sequence. As each state is entered, this program should signal the new state to the second microprocessor then display the state on its own character LCD.
3. A press of any key on the host keyboard, the local 4x4 keypad, or received from the other microprocessor should signal the "emergency services override" as described.
4. The active buzzer should be used to create the audible alarm.
5. The program should continue to process input until the '*'/'E' key is pressed three times in sequence with no other keys pressed in-between. At that point, the program should signal the end of operation to the second microprocessor and shutdown cleanly.

Task #1 Instructions

1. The program that you write second microprocessor should begin by waiting for a message from the first microprocessor to determine which is the current state.
2. Once a message indicating state is received, the E-W lights should be placed into the appropriate configuration and the state number should be displayed on the local character LCD.
3. A press of any key on the console keyboard or the local 4x4 keypad should be sent over Serial1 to the first microprocessor to signal the "emergency services override" as described.
4. The active buzzer should be used to create the audible alarm.
5. The program should continue to process input until the shutdown signal is received from the first program. At that point, the program should shutdown cleanly.

Laboratory Report

Please follow the laboratory report guidelines found earlier in this manual. A single report describing both programs and their interactions should be produced for the group.

Laboratory 5
Pulse-Width Modulation
DC Motor Control and Servo Motor Control

This is an individual assignment.

Background and Related Material

Microprocessor-based Embedded Systems are often used to monitor and control mechanical systems. The primary device for converting electrical energy to mechanical energy is the motor. Embedded systems are capable of controlling a wide variety of different motor types.

All of the motors we will consider operate by routing electrical current through coils of wire to produce magnetic fields which cause a shaft to rotate. Thus, electrical energy is converted to mechanical energy in the form of rotational movement. This rotational movement may be converted to linear movement if necessary. When considering the control of a motor, we will be concerned with several factors. The first is the rotational speed of the motor, expressed in units of revolutions per unit time (e.g. revolutions per second). The second is the position of the rotating shaft, expressed in units of angular displacement from some reference point (e.g. 45º clockwise from the mark). The third is the torque or rotational work produced by the motor, expressed in terms of force times distance (e.g. g*cm or oz*in).

In this laboratory we will explore two types of motor that are commonly controlled using Pulse Width Modulation (PWM).

Typically (brushed, permanent-magnet) DC motors are driven by a single coil which may be energized with either polarity to select clockwise or counterclockwise rotation. At a given input voltage, a DC motor will attempt to spin at a given rate. The actual speed of the motor is dependent on the torque that is required to turn the load. A given DC motor may rotate much faster when unloaded than when loaded. Typically, a DC motor is capable of either high speeds or high torque but not both simultaneously. The variability of a DC motor's speed with the size of its load makes predicting the speed and shaft position difficult, if not impossible, without the use of additional sensor hardware. Microprocessor control of DC motors is often achieved by using the output of position sensors on the motor shaft as feedback to adjust the driving voltage on the motor to achieve the desired motor speed. Adjusting the DC motor input may be accomplished either by adjusting the magnitude of the input voltage using a Digital-to-Analog driver or by using Pulse-Width Modulation to achieve the desired average voltage. We will use PWM.

Servo motors are generally built from DC motors with additional accompanying electronics and gearing. A standard servo utilizes a motor to rotate a shaft to a pre-determined position within a (typical) 180° arc. The servo motor position is selected by sending a 50 Hz (20ms period) pulse train with a variable active-high duty cycle. A 7.5% duty cycle (1.5 ms/20 ms) indicates that the motor should be driven to the center of its arc (the neutral position). Maximum deflections typically occur at 5% duty cycle (1 ms/20 ms) for maximum clockwise

rotation and at 10% duty cycle (2 ms/20 ms) for maximum counter-clockwise rotation. Some servo motors may have a narrower or wider range of input duty cycles. Care should be taken never to try to force a servo motor beyond the natural range of its motion since most have mechanical stops to prevent excess deflection and a motor attempting to push against an immovable stop can easily overheat and fail.

Laboratory Assignment – General Description for DC Motors

The overall goal of this assignment is for you to produce a system capable of controlling a dc motor. The DC motor from the Rexqualis kit is shown below. This has a 2-wire connection. The kit also contains a fan blade which can optionally be fitted on the DC motor which makes its rotation more obvious. The kit will have either a L293 or SN754410 H-Bridge motor driver chip. The DC motor should be powered from the above power module through the L293 or SN754410.



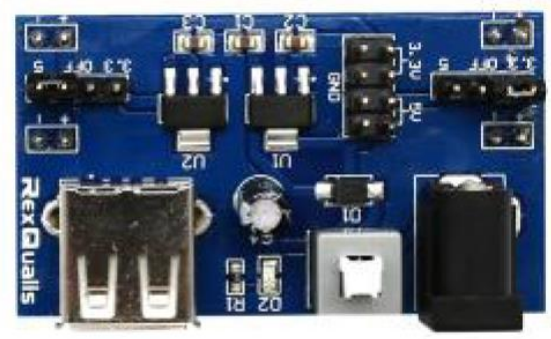
DC Motor Task #1 Instructions

1. The first program that you write is to manage user inputs and determine motor control for a dc motor.
2. When a key is received from either the 4x4 keypad or serial console, interpret it as follows:
    - If the key is “A”, increase the duty cycle by 1 in the range -100 to 100.
    - If the key is “B”, decrease the duty cycle by 1 in the range -100 to 100.
    - If the key is “C”, increase the duty cycle by 10 in the range -100 to 100.
    - If the key is “D”, decrease the duty cycle by 10 in the range -100 to 100.
    - If the key is “0”, stop the motor (set PWM duty cycle to 0)
    - Any other key should be ignored by the motor
3. After the key action is determined, update the DC motor speed accordingly.
4. Additionally, send a suitable message to the console to display the dc motor PWM duty cycle and direction.
5. The program should continue to process input until the ‘*’/’E’ key is pressed three times in sequence with no other keys pressed in-between. At that point, the program should shutdown cleanly.

Laboratory Assignment – General Description for Servo motors

The overall goal of this assignment is for you to produce a system capable of controlling a servo motor. Examples of these are available in the laboratory kit.

Please note that any of the motors considered draw more power than should be taken directly from either of the microprocessor boards. The Rexqualis kits for the Mega contain a power module that would be suitable for providing power to the motors. This module and the associated power supply are shown below and these should be used to supply the power for the motors. Both the Servo motor and the DC motor available from the Rexqualis kit are suited for 5V operation.



The Servo motor from the Rexqualis kit is a small, model-airplane style servo as shown below. This has a 3-wire connection. One wire will be ground, a second will be for 5V, and the third for the PWM control signal. Depending on the manufacturer and model, the ground wire will likely be black or brown, the 5V wire will likely be red, and the signal wire will likely be white, yellow, or orange. While ground and PWM should be connected to the microprocessor, the 5V connection should only come from the above power module.

Servo Motor Task #1 Instructions

1. The program that you write is to manage user inputs and determine servo motor control.
2. When a key is received from either the 4x4 keypad or serial console, interpret it as follows:
    - If the key is “1”, increase the pulse width by 1 in the range 1000 to 2000.

- If the key is “2”, increase the pulse width by 10 in the range 1000 to 2000.
- If the key is “3”, increase the pulse width by 100 in the range 1000 to 2000.
- If the key is “4”, decrease the pulse width by 1 in the range 1000 to 2000.
- If the key is “5”, decrease the pulse width by 10 in the range 1000 to 2000.
- If the key is “6”, decrease the pulse width by 100 in the range 1000 to 2000.
- If the key is “7”, set the pulse width to 1000, the minimum value.
- If the key is “8”, set the pulse width to 1500, the center value.
- If the key is “9”, set the pulse width to 2000, the maximum value.
- Any other key should be ignored by the motor.

3. After the key action is determined, send a update the servo pulsewidth.
4. Additionally, send a suitable message to console to display the servo pulsewidth.
5. The program should continue to process input until the ‘*’/’E’ key is pressed three times in sequence with no other keys pressed in-between. At that point, the program should shutdown cleanly.

Report Preparation

Prepare your report according to the guidelines presented earlier in the laboratory manual and submit it by the due date.

Laboratory 6
Analog-to-Digital Convertors (ADC)
Sensors and Measurement

This is an individual assignment.

Background and Related Material

Microprocessor-based Embedded Systems are often used to monitor and control systems. In order to properly control a system, the microprocessor must be able to sense the state of the system being controlled. Many sensors come in the form of transducers, which convert some physical quantity into a voltage level which can be measured by an analog-to-digital convertor for use by the microprocessor. Such transducers might convert temperature to voltage, or humidity to voltage, or light to voltage, or position to voltage, etc.

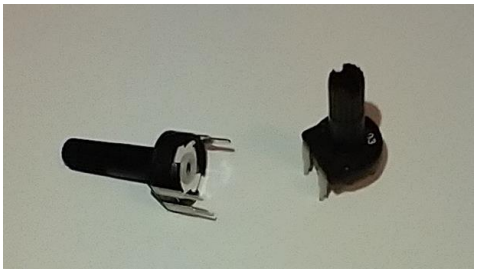
The Arduino Mega 2560 has 16 available built-in analog-to-digital convertor (ADC) pins named “A0” through “A15”. The Mega sampling depth is 10 bits per sample.

Laboratory Assignment – General Description

The overall goal of this assignment is for you to produce a system capable of using ADC to read sensor data and use that data to control physical systems. The physical systems to be controlled are the DC motor and servo motor as used in the previous laboratory. The wiring and power connections, as well as the controls for the motors should be the same as for the previous laboratory.

In the Rexqualis kit, there are several items which provide analog voltage signals which can be read by the ADC pins on the Mega.

The kit contains a potentiometer, which is a resistor whose resistance changes with the turn of a knob. The knob splits the total resistance into two parts and adjusts the split point. This allows the potentiometer to be used to provide any voltage within a range.



The analog joystick module, pictured below, has two analog outputs in addition to ground and 5V connections. The two outputs, labeled VX and VY, which give a voltage proportional to the joystick position along the X and Y axes of the joystick.

The kit contains a Cadmium-Sulfide (CdS) photocell or photoresistor, which is a resistor whose resistance changes with light intensity in a predictable way. By placing the photoresistor into a voltage-divider configuration with a known, fixed resistance, measuring the voltage across the photoresistor allows the resistance and consequently light level to be calculated.

In addition to the parts in the kit, you will be given access to either an LM34, LM35, or MCP9701 which produces an output voltage proportional to temperature. The output of the LM34 is calculated as 10 mV/ºF. The output of the LM35 is calculated as 10 mV/ºC.
The output of the MCP9701 is calculated as 400 mV + 19.5 mV/ºC.

Task #1 Instructions

1. The program that you write is to manage user inputs and calculate useful values from the raw analog data.
2. The program written should read the ADC values for the two joystick axes, the photoresistor, the thermometer, and the potentiometer from the Mega once per second. This is a total of five raw analog-to-digital values in the range 0-1023.
3. Based on these values, the program should make the following calculations and take the following actions:
    - Assume that the center point for the joystick returns value 512 for each axis. Treat this as the origin for a Cartesian coordinate system and scale to a unit circle. This can be accomplished by subtracting 512 from each raw axis value to give a resulting value in the range [-512, 511]. This is then divided by 512 to give a range [-1,1]. Once both X and Y axes are scaled, use their values to calculate a

vector (magnitude and angle) in polar coordinates. The polar vector should be printed on the terminal and character LCD. If the magnitude of the vector is greater than or equal to 0.25, turn on the yellow LED.

- Calculate the ambient temperature from the LM34/LM35/MCP9701 and display it on the terminal and LCD. If the temperature is greater than or equal to 77ºF/25ºC, turn on the red LED display a "HOT" message on the LCD on the Mega then turn on the fan (DC motor). If the temperature is less than 77ºF/25ºC, turn off the red LED and turn off the fan (DC motor).
- Calculate the resistance of the CdS photoresistor and display it on the terminal. If the resistance is greater than 10 kΩ, assume that it is "dark" and turn on the white LED.
- Calculate the voltage on the potentiometer and display it on the terminal. Update the position of the servo such that 0V on the potentiometer means 1000 microseconds for the pulse width and 5V on the potentiometer means 2000 microseconds on the pulsewidth with intermediate voltages generating proportional pulsewidths.
- If none of the other LEDs is lit, turn on the green LED.

4. The program should continue to process input until the '*'/'E' key is pressed three times in sequence with no other keys pressed in-between. At that point, the program should shutdown cleanly.

Report Preparation

Prepare your report according to the guidelines presented earlier in the laboratory manual and submit it by the due date.

Laboratory 7
Serial Peripheral Interconnect (SPI)
Extending GPIO

This is an individual assignment.

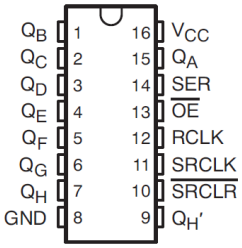
Background and Related Material

Microprocessor-based Embedded Systems are often called upon to coordinate the actions of multiple peripherals. This can easily lead to a shortage of available GPIO pins. The Arduino Mega 2560 is better equipped in this capacity than many microprocessors but any finite resource can be overwhelmed. To overcome this limitation, a number of common serialized communications protocols have been developed to allow for communication using a limited number of pins. Previous labs have used UART and UART-over-USB extensively. This laboratory will use Serial Peripheral Interconnect (SPI). SPI has the benefit of allowing for much faster communication than the typical UART and allowing for expandability with the addition of one pin per device added.

Laboratory Assignment – General Description

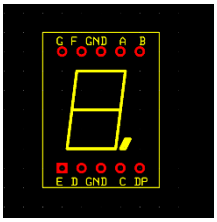
The overall goal of this assignment is for you to produce a system capable of using SPI to interact with peripherals without using an excess of GPIO pins. The system will use the other interfaces developed in the previous labs.

The Rexqualis kit includes a 74HC595 chip which is a serial-input, parallel-output shift register chip. While this chip is not an SPI chip *per se*, it is sufficiently compatible that we can use it with SPI.

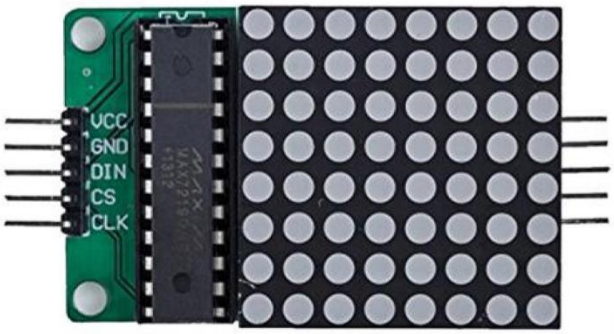


For SPI compatibility:
SCK → pin 11 (SRCLK), MISO → pin 9 ($Q_{H'}$), MOSI → pin 14 (SER), SS → no equivalent, GPIO → pin 12 (RCLK)
Fix OE' (pin 13) as GND and SRCLR' (pin 10) as VCC.

The Rexqualis Kit also includes a 7 segment display. The 7 LED segments plus 1 LED decimal point can be driven easily by the output pins of the 74HC595.



The Rexqualis Kit includes an 8x8 LED dot matrix display with a MAX7219 driver. This allows the control of 64 LEDs individually using only SPI pins. The MAX7219 is interfaced via a SPI interface. The MAX7219 as implemented becomes a matrix LED display driver.



The 2 axis joystick accessed by ADC on the Mega will be needed similarly to previous labs. A potentiometer should be wired to an available ADC pin on the Mega.

Task #1 Instructions

1. The program written should request the ADC value for the potentiometer attached to the Mega. The raw ADC value should be an integer number in the range 0 to 1023. If this is divided by 64, the quotient is a number in the range 0-15 or, equivalently, 0x0-0xF. This should be displayed as 0-9 or A-F on the 7-segment display via 74HC595.
2. The program should also request the X and Y positions of the joystick as analog values. These will be values between 0-1023. If these are divided by 128, the quotients will be numbers in the range 0-7. The resulting value from X should be used to select the row of the 8x8 LED array. The resulting value from Y should be used to select the column of the 8x8 LED array as value 2^Y. The LED at the corresponding coordinates should be lighted and all others should be turned off.
3. The program should also send the full 0-1023 X and Y coordinates to the LCD.
4. The program should continue to process input until the '*'/'E' button is pressed three times in sequence. At that point, the program should shutdown cleanly.

Report Preparation

Prepare your report according to the guidelines presented earlier in the laboratory manual and submit it by the due date.

Laboratory 8
Inter-Integrated Circuit (IIC, I2C, I^2C)
Real-Time Clock, Serial EEPROM

This is an assignment for two people.

Background and Related Material

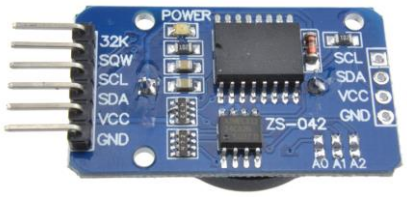
Microprocessor-based Embedded Systems are often called upon to coordinate the actions of multiple peripherals. One method for doing this is communication over a bus. The Inter-Integrated Circuit bus allows up to 128 devices to be connected to a single communication channels using only two pins.

Laboratory Assignment – General Description

The overall goal of this assignment is for you to produce a system capable of using IIC to interact with peripherals without using an excess of GPIO pins. The system will use the other interfaces developed in the previous labs.

The Arduino Mega is not network connected and has no internal real-time clock. An external Real-Time Clock could provide a time base for the Arduino Mega.

The Rexqualis kit includes a ZS-042 board containing a DS3231 Real-Time clock chip and an AT24C32 serial EEPROM memory chip. This is pictured below. The SQW pin can be an interrupt for an alarm time or a periodic clock signal selectable as 1 Hz, 1024 Hz, 4096 Hz, or 8192 Hz.



The ZS-042 board also includes an AT24C32 IIC EEPROM. This will be used to save a time/date marker for later retrieval.

Task #1 Instructions

1. The program written should read the time from the DS3231 RTC and display that time both on the Serial console and the character LCD and then produce a menu of options on the console and poll for a key from the console or keypad.

2. When a key is received, interpret it as follows:
    - If the key is “1”, prompt the user for the time to set the DS3231 RTC.
    - If the key is “2”, update the time in the RTC forward by 1 hour.
    - If the key is “3”, update the time in the RTC backward by 1 hour.
    - If the key is “4”, update the time in the RTC forward by 1 minute.
    - If the key is “5”, update the time in the RTC backward by 1 minute.
    - If the key is “0”, update the seconds in the RTC to be 0.
    - If the key is “A”, save the current date and time into the AT24C32 EEPROM.
    - If the key is “B”, read the saved date and time from the AT24C32 EEPROM and display it on the console with an appropriate message.
    - Any other key should be ignored by the clock.
3. The program should continue to process input until the ‘*’/‘E’ key is pressed three times on the keypad in sequence with no other keys pressed in-between. At that point, the program should shutdown cleanly.

Report Preparation

Prepare your report according to the guidelines presented earlier in the laboratory manual and submit it by the due date.

Laboratory 9
Integrated Application
Weather Station

This is an assignment for two.

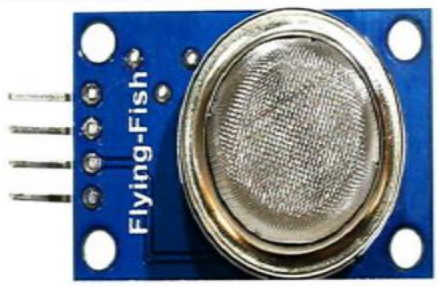
Background and Related Material

The previous laboratory assignments have focused on adding a few peripherals and their associated communications and controls. This laboratory is going to focus on utilizing multiple of these peripherals to create a useful application.

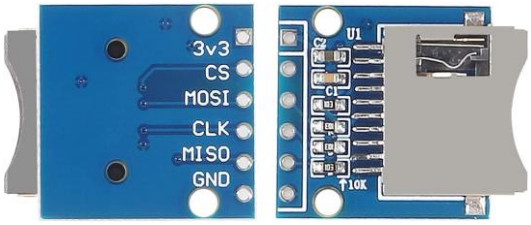
Laboratory Assignment – General Description

The overall goal of this assignment is for you to produce a system capable of acting as a weather station with data gathering capabilities. This will be accomplished across two microprocessors.

Previous laboratories introduced a temperature sensor, one of LM34, LM35, or MCP9701. Similarly, the previous labs introduced the CdS photocell to measure light. The Rexqualis kit contains other sensors appropriate for a weather station. For example, the MQ-2 is used to measure air quality. Its output is an analog voltage proportional to the pollutants in the air.



To create a "weather station" you are going to create a system which will gather weather-related data and save that data onto a secure digital card. You will be given access to an SD card and SD-card interface similar to the one pictured below. This will allow access to an SD Card via SPI.



Task #1 Instructions

1. The first program that you write is for the microprocessor which will gather the raw data.
2. This microprocessor should have a DS3231 RTC clock attached. The DS3231 RTC should be set to create a 1 Hz square wave which will trigger an interrupt to indicate when samples should be taken.
3. Data read in response to the interrupt should be sent to the second microprocessor over Serial1 to be stored in a file on the SD Card.
4. The main loop of the program should wait until a signal from the 1Hz interrupt arrives then …
    - Read the temperature, light level, and air quality
    - Write a new line to Serial1 with data in the following format
        - Date, Time, Temperature in Fahrenheit, CdS resistance for light level, air quality voltage
    - If 0V <= MQ-2 air quality sensor voltage <= 0.5V, light the green LED and turn the other LEDs off
    - If 0.5V < MQ-2 air quality sensor voltage <= 1.0V, light the yellow LED and turn the other LEDs off
    - If 1.0V < MQ-2 air quality sensor voltage <= 5.0V, light the red LED and turn the other LEDs off
    - The loop should repeat each time the 1 Hz trigger happens
5. The program should continue to process input until the '*'/'E' key is pressed three times on the keypad in sequence with no other keys pressed in-between. When this occurs, a message consisting of a line with "-1" for the date should be sent to the second microprocessor to signal the end of data. At that point, the program should shutdown cleanly.

Task #2 Instructions

1. The second program that you write is for the microprocessor that will be interfacing the SD Card and saving the data coming from the first microprocessor.
2. The program should begin by either opening or creating a file named "weather.csv" in the root directory of the SD Card. If the file previously existed, new data should be appended to the end of the file.
3. The main loop of the program should wait and expect data to be sent one line at a time with each line formatted as follows …
    - Date, Time, Temperature in Fahrenheit, CdS resistance for light level, air quality voltage
4. If the date value is not "-1", the line of data is to be added to the file.
5. If the date value is "-1", the file should be closed and the program may end.

Task #3 Instructions

1. After capturing data into a "weather.csv" file on the microprocessor and closing the file, transfer the SD card to the PC and open the CSV file in Excel.
2. Select the data and use it to create a graph for each column of data.

3. Save the result as an Excel file which will be submitted along with your lab report.

Report Preparation

Prepare your report according to the guidelines presented earlier in the laboratory manual and submit it by the due date.