Design, Testing, and Development of an Internal Combustion Engine PID Throttle Controller.

by

Michael Burlingame

BioResource and Agricultural Engineering

BioResource and Agricultural Engineering Department

California Polytechnic State University

San Luis Obispo

2014

Interesting project.

TITLE	:	DESIGN, TESTING, AND DEVELOPMENT OF AN INTERNAL COMBUSTION ENGINE PID THROTTLE CONTROLLER
AUTHOR	:	Michael Burlingame
DATE SUBMITTED	:	June 13th, 2014

Dr. Andrew Holtz
Senior Project Advisor

Andrew J. Holtz
Signature

6/13/14
Date

Art MacCurdy
Department Head

Signature

7/7/14
Date

ACKNOWLEDGEMENTS

I would like to thank Dr. Andrew Holtz for his guidance and support on this project. I would also like to thank Virgil Threlkel for his support of the encompassing Horsapillar project, of which is necessary for this project to be relevant. I would like to thank Cal Poly's BioResource and Agricultural Enginnering Department for its financial support on this project.

ABSTRACT

This senior project discusses the design and testing process of a 32-bit PID microcontroller capable of starting and controlling the throttle position. The purpose of this project is to complete one portion of a much larger department-funded vehicle project for BioResource and Agricultural Engineering at Cal Poly, San Luis Obispo. The engine chosen for this application is a Kohler Magnum 18 horsepower two-cylinder horizontally opposed four-stroke engine. Engine speed information is gathered using the pre-existing fixed-timing magneto ignition.

The process of conditioning a high-voltage ignition pulse to a fixed pulse width, digital signal is described in detail. In addition, fail-safe circuits were developed for throttle control override and engine over speed governing. Oil Pressure and Temperature monitoring circuits were developed for additional engine-monitoring purposes. A small diagnostic screen was incorporated in order to provide on-board troubleshooting during later testing.

DISCLAIMER STATEMENT

The university makes it clear that the information forwarded herewith is a project resulting from a class assignment and has been graded and accepted only as a fulfillment of a course requirement. Acceptance by the university does not imply technical accuracy or reliability. Any use of the information in this report is made by the user(s) at his/her own risk, which may include catastrophic failure of the device or infringement of patent or copyright laws.

Therefore, the recipient and/or user of the information contained in this report agrees to indemnify, defend, and save harmless the State its officers, agents, and employees from any and all claims and losses accruing or resulting to any person, firm, or corporation who may be injured or damaged as a result of the use of this report.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

With an industry in pursuit of reducing production costs, meeting stricter emission requirements, and building a better product, computer control has moved into being an essential role for agriculture equipment. More specifically, hydraulics for loader buckets, implements, accessories, drivetrains, are moving away from mechanical controls, spool valves, and stack pumps, and utilizing computer controls to actuate digital variable displacement pumps, pulse-width modulated proportioning valves, and monitoring system pressures and temperatures. Along with many competitors, one of these companies is Danfoss Group Global. Danfoss has introduced into industry the Plus+1 microcontroller system; a system designed specifically for mobile control.

Since robust microcontrollers, such as the Plus+1, are of significant importance in the agriculture industry, Cal Poly's BioResource and Agricultural Engineering (BRAE) department is interested in providing instruction and experience to its students in becoming fluent in microcontroller programming and utilization. At present many projects are under way in the BRAE dept. to develop microcontroller learning tools to be used in the classroom laboratory. One specific project is the development of a small

Figure 1. Horsapillar concept vehicle is shown.

skid-steer vehicle with a computer-controlled hydrostatic drivetrain, titled "Horsapillar" (Figure 1).

On "Horsapillar" the Plus+1 module will be designed to monitor user inputs from both a DP600 display and a JS6000 joystick (Danfoss Products). The user will input speed and direction, while the Plus+1 microcontroller will process those inputs and determine the necessary displacement of each pump and required engine speed. Since the engine that will be used is smaller scale than what would be commercially available for this application, the objective of this senior project is the design, construction, testing, and evaluation of a microcontroller system that will monitor engine inputs and make necessary decisions on throttle position.

For this specific project, the engine used will be a Kohler Magnum 18 two-cylinder horizontally-opposed 4 cycle gasoline engine. This specific engine was chosen because of its availability as a BRAE dept item.

In addition, this particular engine model came with a 9T 16/32 DP SAE-A flange (Appendix B) directly to a light duty hydraulic pump without any additional coupling hardware.

The Magnum 18 is an extremely basic industrial engine. It uses a waste-spark magneto-type ignition system, a stator-type charging system (similar to that used on small motorcycles), and a single-barrel side draft carburetor with an electric choke. The original fuel system regulated engine rpm using a mechanical governor and spring controlling throttle position.

The engine management system must have several attributes to work successfully. This system shall be capable of sending and receiving information to the Plus+1 controller module via a CANBus serial connection. The unit should be rugged and complimenting circuitry should be weather protected. The microcontroller should be capable of being quickly reprogrammed while installed for ease of testing and lab demonstrations. The power source will be the tractor's primary 12.6V battery in conjunction with the charging system. This implies the system must be capable of accepting a somewhat low quality power source, filtering and regulating it where necessary. Ideally the engine should be able to compensate for varying engine loads, in addition to understanding when a desired engine speed is not possible based on conditions, and transmit that information accordingly via CANbus serial communication.

In addition to the sensors and actuators necessary to control engine speed, the engine management should also be able to transmit engine condition information such as: engine operating hours, engine temperature, and engine oil pressure. This information may be vital to the operator to prevent engine damage in event of a problem.

LITERATURE REVIEW

Research was conducted on both prior design projects, information available on mobile control with the Plus+1 system, and on engine management systems.

The basic components behind any automatic process include 3 essential categories: Sensors and Transmitters; Controllers; and a Final Control Element. From these components three essential tasks are performed in any control system: Measurement, Decision, and Action (Smith and Carripro 1997).

The overall goal of automatic process control is to "adjust the manipulated variable to maintain the manipulated variable to maintain the control variable at its set point in spite of distrubances" (Smith and Carripro 1997). Control of the set point occurs in two fashions: regulatory control of disturbances for a constant set point, and servo control of disturbances for a profile set point (Smith and Carripro 1997).

There are two methods in correcting for disturbances: The first method, feedback control, is applicable to almost all situations. When the control value deviates from the set point, the microcontroller determines corrective actions for control elements to bring the control variable back to the set point. A disadvantage to this process is that the control variable will deviate before any action is taken. In some cases this may not be acceptable.

An alternative to the prior is feed-forward control. Feed-forward control has sensor or transmitter devices ahead of the process to predict necessary actions to take to keep the control variable at the set point. The disadvantage of feed-forward control is the complexity in predicting necessary actions, and some systems may not have adequate latency to allow for changes to occur before the control variable has changed.

In the interest of better understanding areas of greater yield within a field, modern data acquisition technology has been adapted to a rice combine in hopes that it will bring new light into precision agriculture (Kin et al. 2011). A Dewe-2010 PC data acquisition system was used with the following inputs: Differential Global Positioning System (DGPS)(Trimble AgGPS 132 DGPS, Grain Flow Sensor, Grain Moisture Sensor, Fuel Flow Sensor, Grain Loss Sensor, Radar Velocity Sensor, Drive Axle Shaft Torque Transducer, Theoretical Ground Speed Sensor, Ultrasonic Displacement Sensor, Header Position Sensor, and a Tilt Sensor. How sensors are connected to the Dewe-2010 module is well organized with a chart that divides up sensors that will change in voltage, frequency, or serial data (connected to a DAQP-Bridge). The article describes ways in which custom sensors were fabricated, such as a grain flow sensor. The flow sensor was created by developing a plate within the grain chute that sensed forced applied by the incoming flow of grain. The force was measured using a Wheatstone Bridge load sensor attached to the back of the plate. To quantify the data, the sensor was calibrated by feeding known quantities of grain on a custom conveyor into the chute.

Another article, published in 2007, discusses the need, design, and implementation of a cruise control with model-based predictive control on a combine harvester (Coen et al. 2008). In interest of reducing noise and air pollution, a control system was developed to minimize engine speed for the various

changing tasks done by the combine: There may be times when an operator will be driving slowly across a field, but running the engine wide open throttle (WOT) because a high enough flow is needed to run equipment (but the additional horsepower may not be needed.) There are also times where the operator may need to vary combine speed when traveling from field to field, but not require that the harvesting components are running. The system used a National Instruments PXI system connected to a CANbus serial line for computations. The various inputs included: driver controls, engine speed, and pump current (hydraulic pump flow is controlled by average electric current to an actuator), while modeling considers, under multiple circumstances, actual pump current and engine speed. In order to minimize the amount engine speed is changed, a computerized system of "penalties" are used when changes must be made that are undesirable to quiet and fuel-efficient operation. Results yielded a non-linear model relationship between pump current and engine speed with respect to vehicle speed.

In 2008, a master thesis was submitted by Charles Combs outlining the steps taken in adding an embedded java microcontroller to automate the control of the Cal Poly Rose Float (Combs 2008). Main reasons for upgrading from the current PID control included operator ergonomics, lack of versatility with relay control logic, and the addition of complexity for more sophisticated float designs. The rose float powerplant is a 7.5 liter Ford 8 cylinder gasoline engine converted to run on LPG (Liquid petroleum gas). The original throttle control used an electrical solenoid to actuate a throttle plate. The engine powered two pumps: the main is a Sauer Sundstrand variable displacement pump. The auxiliary pump also driven by the engine is a Rexmann Roth fixed displacement pump.

PROCEDURES AND METHODS

Mechanical Linkage

Mechanical Linkage Design

The initial design began in adapting a servo mechanism to the existing throttle lever. There were no detailed drawings regarding this engine available, so it was necessary to make precision measurements along with some estimation of existing components(Figure 2), and then develop a representative model in SolidWorks that would suffice in determining the layout of mounting brackets (Figure 3).

Figure 2. Key points on the manifold were measured as shown using the digital readout from a vertical mill.

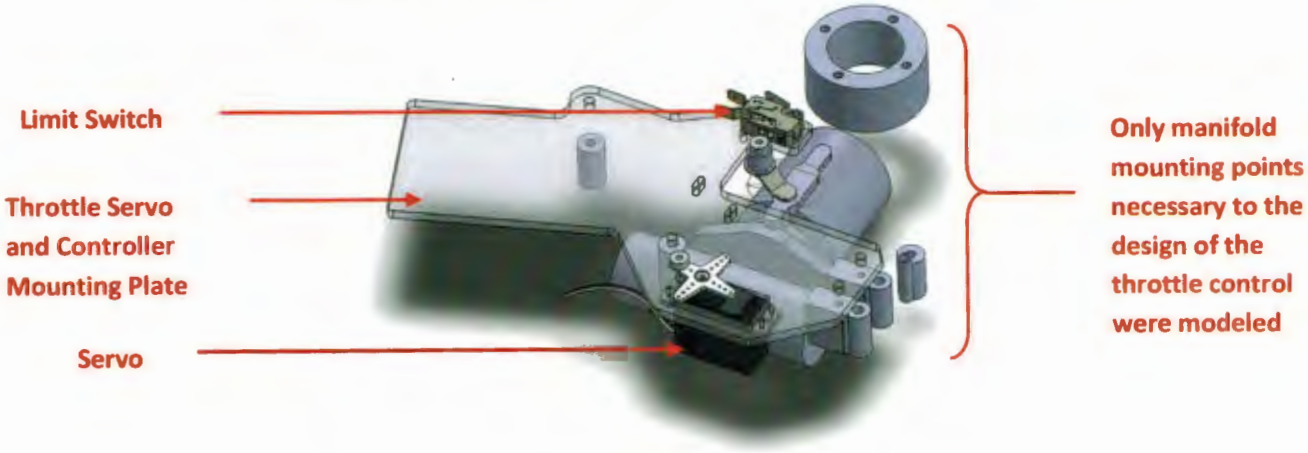


Figure 3. The model drawn using Solidworks from the measurement method in Figure 2 is shown.

A panel was designed that mounted a standard-sized hobby servo at suitable location to connect to the carburetor throttle lever without obstruction. Additional space was incorporated into the aluminum panel to allow a rectangular three-inch by five-inch mounting area for electronics. The panel was constructed from 5052-H36 Aluminum, and some spacers had to be made from 6061-T6. Since the manifold it was fastened to is cast aluminum (assumed to be 356, 242, or similar), no precautions for galvanic corrosion were taken.

The original mechanical governor on the engine was no longer functional, so the majority of its components were removed. The throttle linkage connecting the servo to the throttle plate was assembled using hobby-aircraft ball linkages and hobby-vehicle OEM-specific stainless steel turnbuckle (Appendix C).

Since the upper and lower limits of the throttle position would have to be known for electronic control; adjustable momentary limit switches, along with a contact roller, were incorporated into the mechanical design (Figure 4)

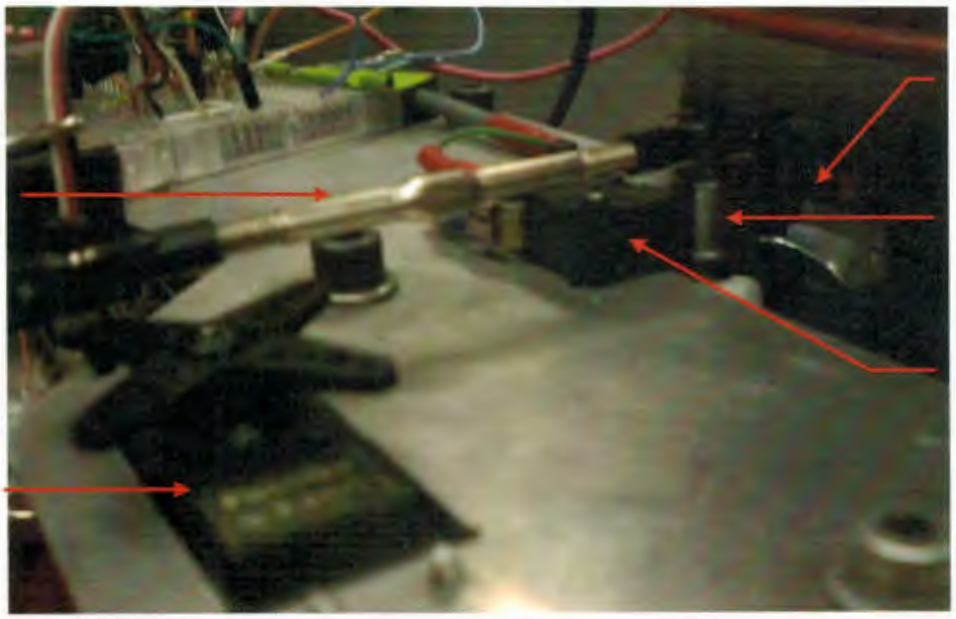


Figure 4. The completed throttle linkage is shown.

The actuator chosen to drive the throttle linkage was a Hobby Servo. This servo was chosen for its common availability as an industry standard, and also exceeding the torque and travel requirements for this application (Appendix D).

Mechanical Linkage Testing

For initial linkage testing, a manual servo controller from ServoCity.com (Figure 5) was used to verify usable electronic range of the servo—the servo is physically capable, when there is no electrical power, of rotating a greater angle end to end than what it will do when receiving power and a position signal, so this needed to be verified. A simple DMM was used to test continuity of the limit switches in order to adjust their position relative to the carburetor throttle plate position. An explanation of the electronic calibration process is covered in *Engine Throttle Control and Cranking Sequence*.

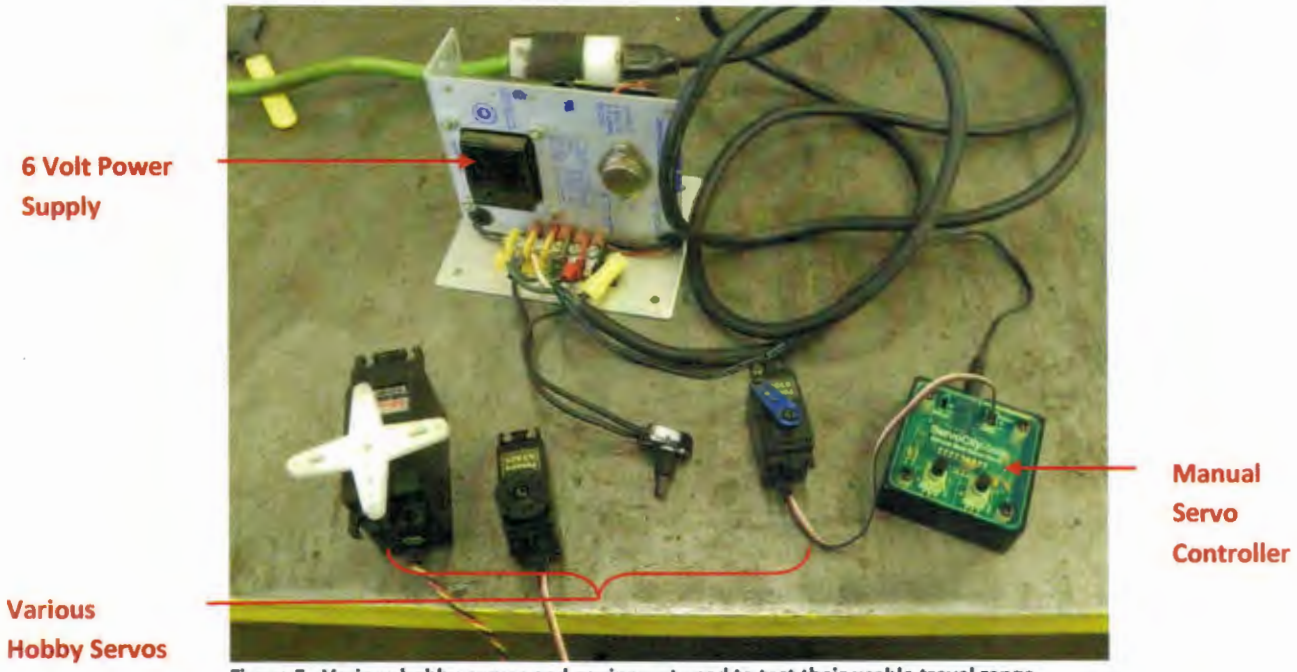


Figure 5. Various hobby servos and equipment used to test their usable travel range.

Electronic Components

Microcontroller Selection

Multiple low-cost microcontrollers were considered for this project: Arduino Uno, Arduino Mega 2560, NetDuino 2, DFRduino MEGA2560, and ChipKit Max32. All are based on the original Arduino platform; it was decided to utilize the ChipKit Max32 based on its cost, customer reviews, and capability. Refer to Appendix E for additional Max32 specifications.

Engine Speed Signal

The Kohler Magnum uses a fixed timing, waste-spark magneto ignition system. It was decided to use this magneto pulse as the source of engine speed signal for the microcontroller. The magneto "kill wire" outputs a fraction of the energy sent through the complete secondary winding. Initial testing, using a

voltage divider circuit (Figure 6) gave ballpark values of a 100V spike at idle, and possibly reaching as high as 250V at higher engine speeds. The spike at each ignition pulse showed several trailing cycles to dissipate the energy produced from the magneto. Since digital circuitry is designed to accept signals of 5V or less in most cases, it was very apparent that signal isolation and conditioning would be necessary in order for a microcontroller to make clear, unflawed speed measurement.

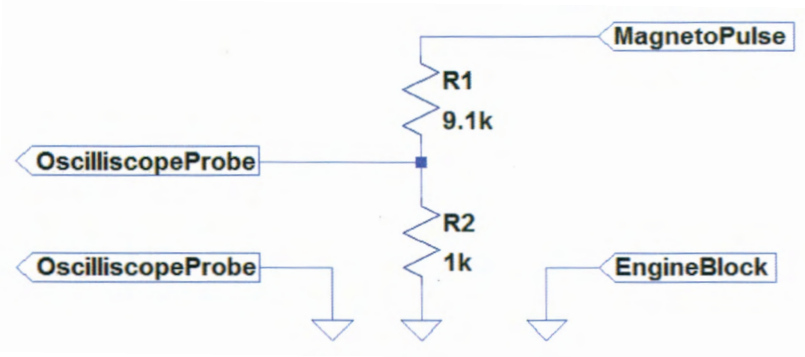


Figure 6. The circuit constructed to measure the magneto "kill wire" pulse on an ascilliscope is shown. Note that the probe used had an additional 1:10 signal ratio.

Net signal ratio calculations are as follows:

$$V_{magneto}\left(\frac{1K}{1K+9.1K}\right)(1{:}10\ Oscilliscope\ Probe) \approx \frac{V_{magneto}}{100}$$

Some constraints of designing this conditioning circuit included:

- Optical Isolation (to prevent the possibility of high voltage reaching critical components)
- Solid State
- Signal should be a square wave with a fixed pulse width
- Provide a stable signal through the entire rpm range of the engine.

The conditioning circuit block diagram was then designed to meet these parameters (Figure 7)

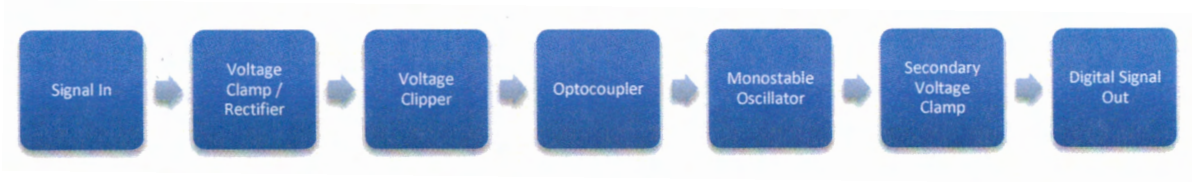


Figure 7. A block diagram converting a magneto "kill wire" pulse to a digital signal is shown.

An initial design of the schematic was done in LTSpice (Figure 8) and then constructed on a solder-less breadboard (Figure 9). Initial testing was done by observing signal output on a Phillips PM3337 Oscilliscope. Proper pulse width and period were observed. One noted error was that the output voltages did not quite match the required high and low voltages required by the microcontroller. Low voltage signal showed only reaching approximately 1Volt. It was determined that this is due to the comparator reaching its "rail" voltage when swinging low – a limit as to how close the comparator output can get to the positive or negative supply voltages. The circuit was changed such that the output voltage was clamped with a 3.3V zener diode, rather than clipped. Testing proved that this adequately pushed the voltage to zero at low state, and approximately 3V at high state (refer to Appendix E for microcontroller digital input requirements).

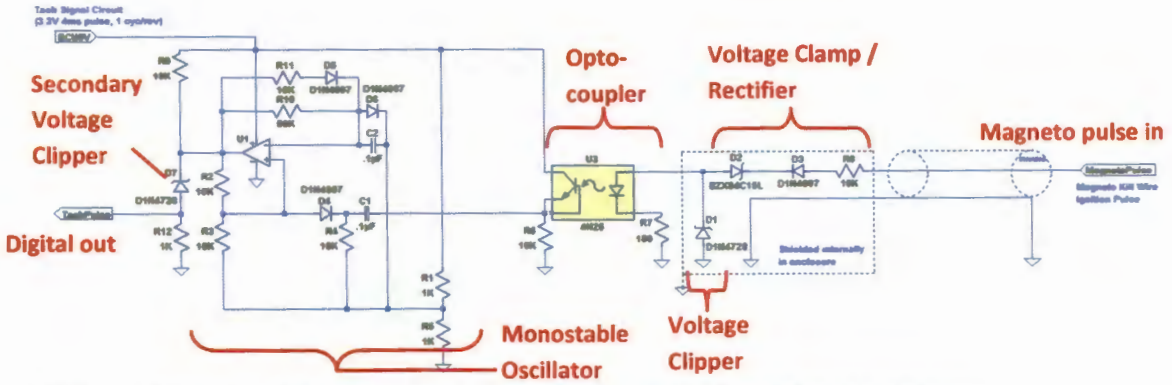


Figure 8. A schematic of the circuit conditioning the magneto pulse to a 3.3V digital signal is shown.

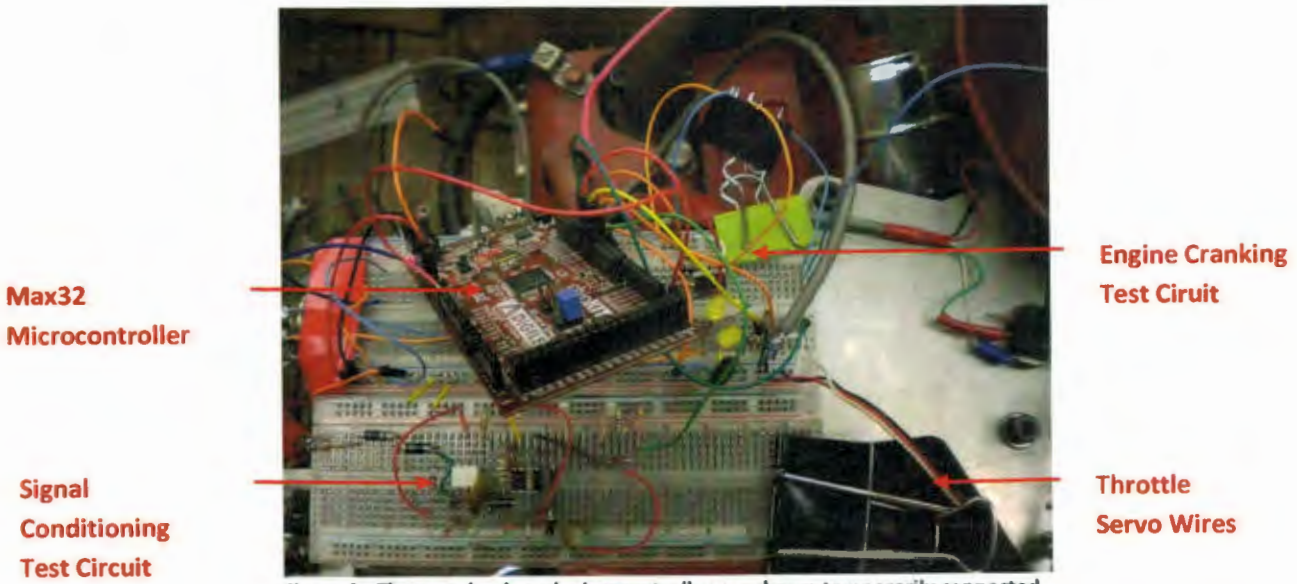


Figure 9. The test circuit and microcontroller are shown temporarily connected to verify functionality of filter, servo control, and cranking circuit.

At this point the signal-conditioning circuit was attached to the microcontroller: It was verified that the microcontroller was accurately reading the signal by writing a simple program that output the engine's calculated RPM signal through the serial port. Several methods of calculating this were considered:

1. Have a counter, based on the microcontroller's internal clock, count the number of pulses in a given sample time. The sample is a fixed-length loop.
   Pros- This method inherently averages several RPM cycles making the resulting value an accurate representation of the engine speed
   Cons- This method would require that the microcontroller pause for a substantial amount of time in order for the engine speed resolution to be at all useful. Refer to equations 1, 2, 3, 4, and 5.

$$Maximum\ Engine\ speed \approx \frac{3600\ rev}{min} \quad (1)$$

$$For\ a\ resolution\ of\ \frac{100\ rev}{\min cycle} \quad (2)$$

$$\frac{3600\ rev}{min} \times \frac{\min cycle}{100\ rev} = 36\ cycles \quad (3)$$

$$\left(\frac{3600\ rev}{min}\right)^{-1} = \left(\frac{60\ rev}{sec}\right)^{-1} = \frac{0.0167\ sec}{rev} \quad (4)$$

$$\frac{0.0167\ sec}{rev} \times (36\ cycles + 1) \times \left(\frac{1\ cycle}{1\ rev}\right) = 0.6166\ seconds \quad (5)$$

   At minimum, with a resolution of 100 RPM, the program would have to pause in a sample loop for 0.6166 seconds. The microcontroller will have to react considerably faster than this for any practical use.
2. Have the engine speed signal use an interrupt-capable input that will, at any point, interrupt the program and place a time stamp on a variable representing the leading edge of the signal. The program would compare the period between the current time stamp and the previous and determine the RPM based on period length
   Pros- This method will yield fairly accurate results at fairly good resolution.
   Cons- using interrupts can cause errors in a program, making it less stable. If the microcontroller was to get an interrupt at the moment it was sending or receiving serial information, the value would be skewed because the interrupt would go into que, until the current interrupt (serial communication) finished.

3. Have the engine speed monitored using a measurement of period length (as done in option 2). However, let the microcontroller run through an entire program cycle, and run through a small loop waiting until the next cycle pulse. The engine speed would then be a function of the last time stamp and current.
   Pros- It allows period length measurement without the use of interrupts.
   Cons- Serial communication interrupt may cause the engine to "miss" a signal pulse. Additional code will have to be added to compute "realistic" changes in RPM; some RPM computations may require a "do-over". Also, the overall length and complexity of the program is limited, to where it must be able to complete an entire loop cycle in less than 0.0167 seconds. At this early stage in the design process there is no guarantee that this will be possible, with future upgrades, etc.

4. Have the engine speed monitored using a measurement of period length (as done in option 2 & 3). However, the microcontroller will have a loop that monitors three or four cycle pulses, and averages the engine speed between them (then returning to the rest of the program loop).
   Pros- It allows period length measurement without the use of interrupts. Also, averaging three to four cycles each time a calculation is made will average the difference between left and right cylinders if there is a power imbalance between them.
   Cons- This method could also be interrupted by serial communication, and will also require additional code to apply logic to the engine rpm computation. In addition, at low RPM the loop may take .24 seconds to compute engine RPM. A compromise to this may be to only averages two cycles, at low engine speeds, taking only 0.12 seconds

Of the methods listed, the fourth was chosen as the option with the best compatibility. After extensive troubleshooting, refer to Appendix M for the open source code.

Testing of the program consisted of attaching the prototype circuit to the microcontroller, and writing a program that, using the specified method above, output engine speed information via USB connection to a pc computer. The serial port data was observed using the microcontroller compiler's built-in serial port monitor (Figure 10).

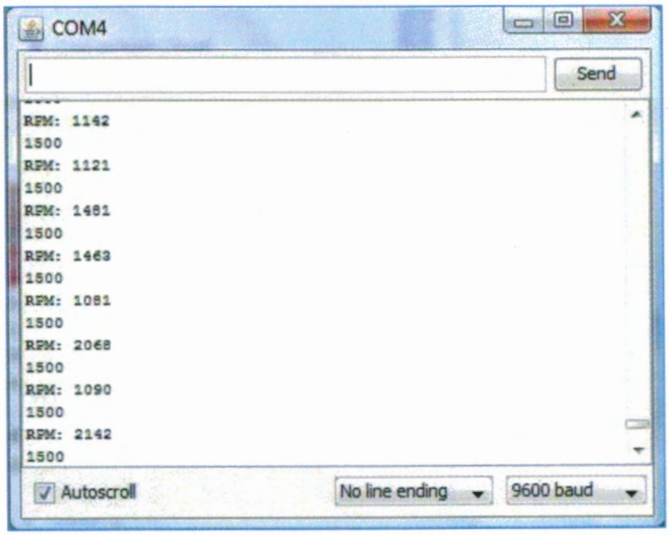


Figure 10. A screen capture of the MPIDE Serial Monitor is shown displaying engine speed set point and current measurement. The value to the right of the "RPM" is measured engine speed, and the value on lines between "RPM" is the set point. Note that this initial test showed some instability from unsoldered connections on the test circuit.

Engine Throttle Control and Cranking Sequence

The next test was to attach the throttle actuator to the microcontroller (in addition to the engine speed signal circuitry). This required no additional components for the initial test. However, after several runaway tests, the following observations were noted:

- The throttle servo current draw exceeds the output of the microcontrollers internal 5V supply, therefore an external 6V voltage regulator is necessary (the servo will run on 4.8volts or greater)(Appendix D). Powering the servo with the microcontroller's internal regulator caused enough of a voltage drop that considered the voltage drop on the signal wire as a pulse, thus running away with its own current draw spikes.
- An additional "default" throttle position circuit is necessary, so if there is any "hang-up" in the microcontroller, it will default to a pre-calibrated setting.
- The servo, as it is mounted on the engine, is configured such that if the servo loses signal, it defaults to wide open throttle. It was decided the best precaution for this problem (as well as a fail safe for other possibilities) is to add an electronic governor.
- Output pins on the microcontroller default to high until the microcontroller program has initialized. This can lead to the engine cranking prematurely or a throttle swinging beyond its designated range set by the limit switches. The solution for this was to include a sub-main relay that only powered the microcontroller. Once the microcontroller has initialized, it will then turn on the main relay (powering all other devices) with the necessary microcontroller outputs set.

The test program requires a "limits scan"- where prior to the engine running, it slowly moves the throttle toward both extents, taking note of when the limit switches are triggered. This is to ensure proper scaling of extreme low positions to extreme high positions, without going beyond what the throttle plate was mechanically designed to do. The specific code for the limit scan is shown in Appendix M.

Once the throttle position has been calibrated, it is set to a predetermined value during cranking. A throttle position slightly off-idle appeared to be the most effective while cranking. Cranking was determined to be a 3 second trigger, set by the microcontroller. If the engine did not start after a 3 second cranking pulse, it would wait for a second trigger from the operator (Figure 11).

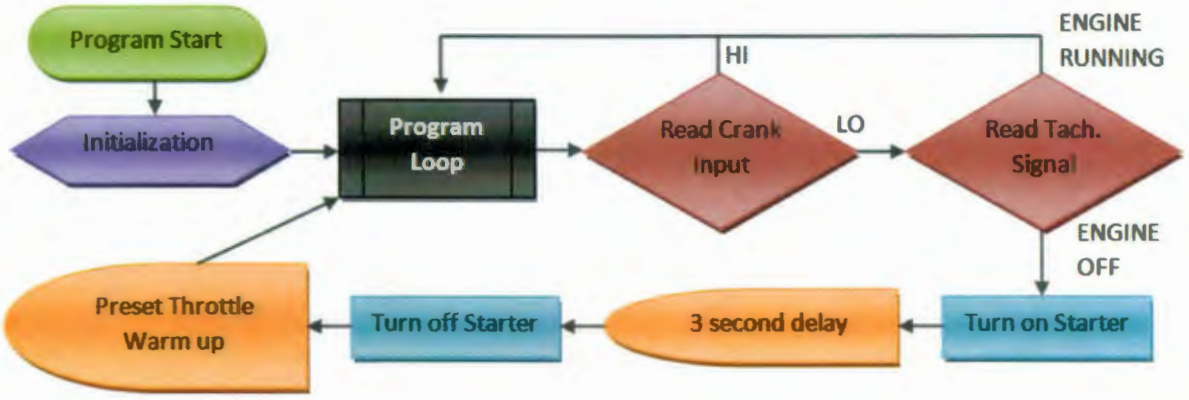


Figure 11. Flow Chart of Cranking Algorithm is Shown

Once running above 1000 RPM for a preset start up time (in seconds), the program throttle position switches from the cranking/warmup-setting to serial input or manual values. After some debugging, the program showed success in receiving an analog potentiometer throttle position signal, then scaling the output to move the servo within the constraints. Refer to the flow chart in Figure 12: The serial input, where a specific RPM value was requested via USB connection, required some constraints to keep the engine speed under control.

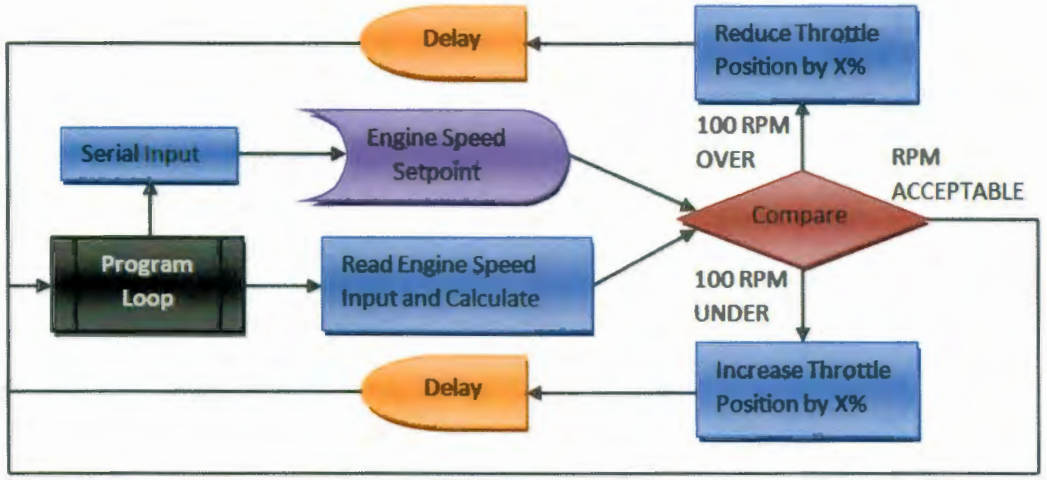


Figure 12. The basic throttle correction algorithm used in testing and development is shown.

The algorithm shown is only the first step in engine control. Though it was able to produce quick, working results, since it was only a fixed error correction factor, it showed some signs of oscillation. At a later point a PID function is implemented to eliminate unstable oscillations and smooth engine response.

Development of Safety Circuitry

As outlined in Engine Throttle Control and Cranking Sequence, several additional circuits are necessary to protect the engine in event of microcontroller failure.

Throttle Fault

A circuit was developed that allows the throttle to default to a preset value, should the microcontroller crash, or information going into to the microcontroller is not adequate to determine engine speed. The circuit is designed around a common 555 timer, and some logic gates (Figure 14). The throttle fault is triggered by either the microcontroller computer or manual override. Refer to Figure 13 for a flow chart explaining when the override circuit will be used.

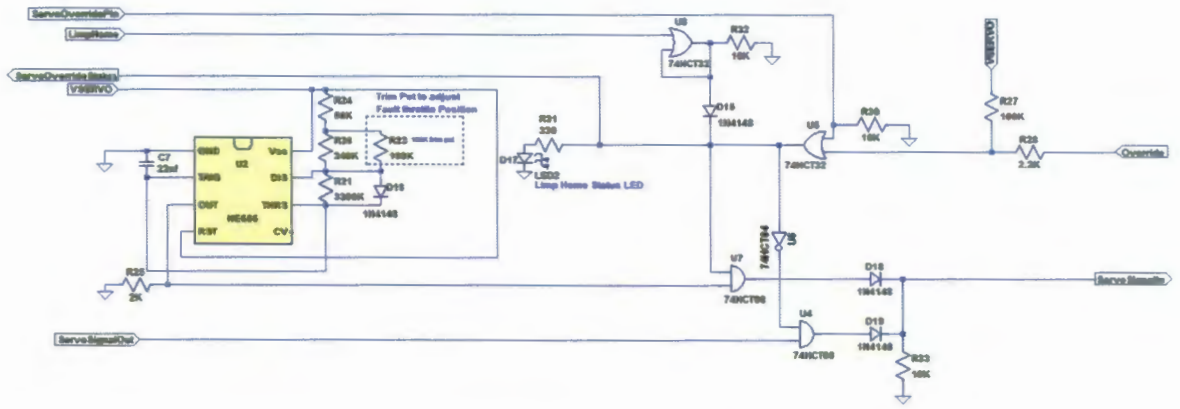


Figure 14. The schematic of the throttle fault circuit is shown.

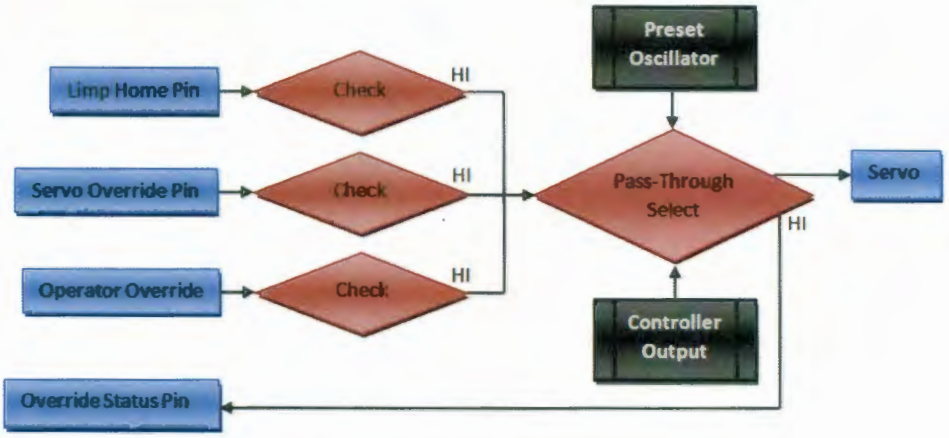


Figure 13. A flow chart of the throttle fault circuit is shown.

Governor

A governor circuit was developed based on an LM2907-8 Frequency to Voltage Converter integrated circuit. This component was chosen due to its frequent use in automotive tachometers, and its ability to be used as a frequency-controlled trigger with hysteresis (Figure 15). Refer to the flow chart in Figure 16 for circuit operation.

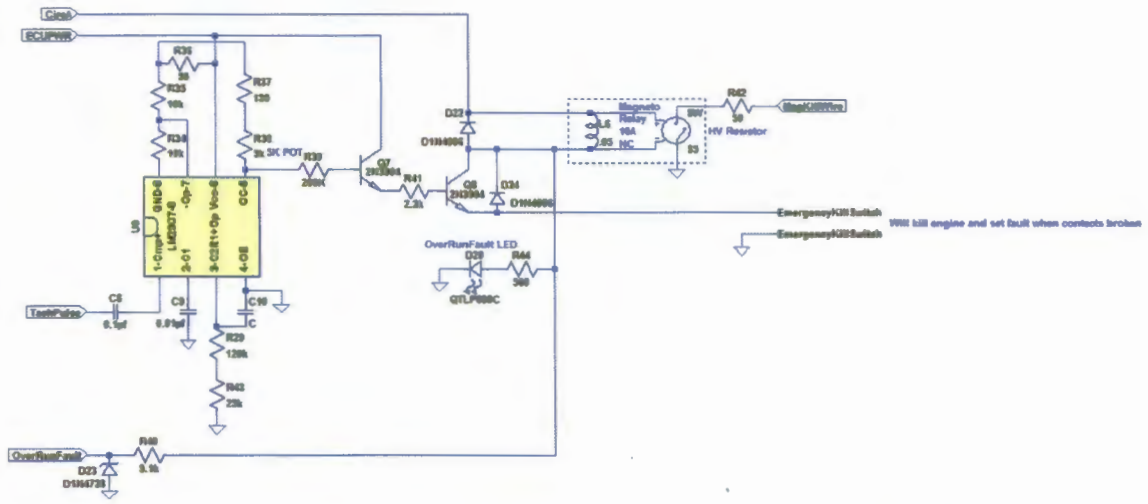


Figure 15. The schematic of the electronic governor is shown.

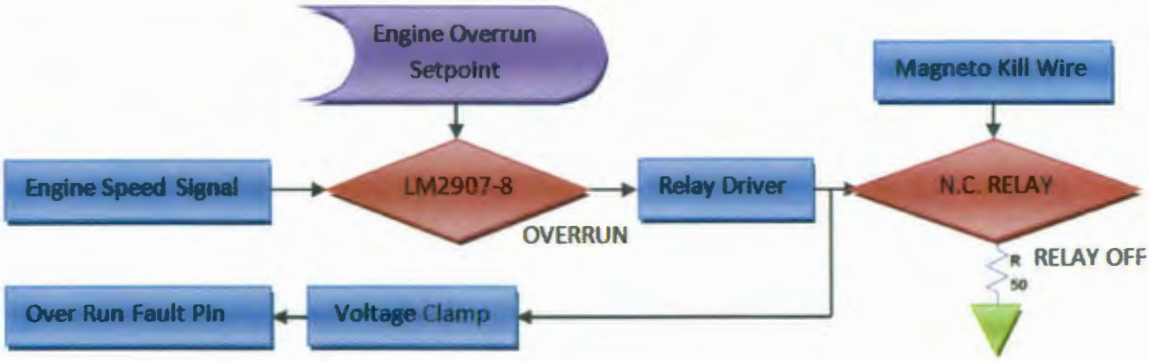


Figure 16. A flow chart explaining the electronic governor operation is shown.

Emergency Kill Switch

An emergency kill switch has been incorporated into the relay used to ground the magneto kill wire (used for microcontroller control as well as speed governor). When pushed, the emergency kill switch will open the relay driver's connection to ground, turning off the relay (thus grounding the magneto kill wire).

Oil Pressure Sensor

Additional information for the operator and for data logging was considered useful. The following inputs with a brief description were included in the microcontroller's information.

The Kohler Magnum 18 engine comes with 1/8 inch FNPT ports for adding an oil pressure sender or switch. Since the oil flows past this point during engine operation, it was decided that this would be an ideal location to measure oil temperature as well as pressure (Figure 17).

Figure 17. The Available Oil Port for Pressure and Temperature is shown.

.A resistive-based sensor was chosen for measuring oil pressure: Echlin OP6676. This part is designed to meet specific General Motors Corporation OEM specifications. It was chosen based on its availability, size, and cost (Appendix C). Since aftermarket companies seldom provide technical information to the consumer, it was not possible to obtain a data sheet or a response curve. In lieu of this, the oil pressure sensor resistive response was mapped through testing.

Oil Pressure Sensor Test Procedure

A test apparatus was constructed of spare components readily available in the Cal Poly BRAE facilities. Key components included:

1. Pressure Regulator/Water Separator Unit
2. Male Air Hose Adaptor
3. Analog Pressure Gauge ranging 0-100psi
4. Schrader Valve to 1/8" MNPT Adapter
5. 1/8" FNPT Tee
6. Handheld Digital Tire Pressure Gauge, accurate to ±0.1psi
7. Pressure Sensor and Electrical Connector
8. Fluke 179 DMM

Refer to Figure 18 for test apparatus diagram and Figure 19 for test apparatus setup.

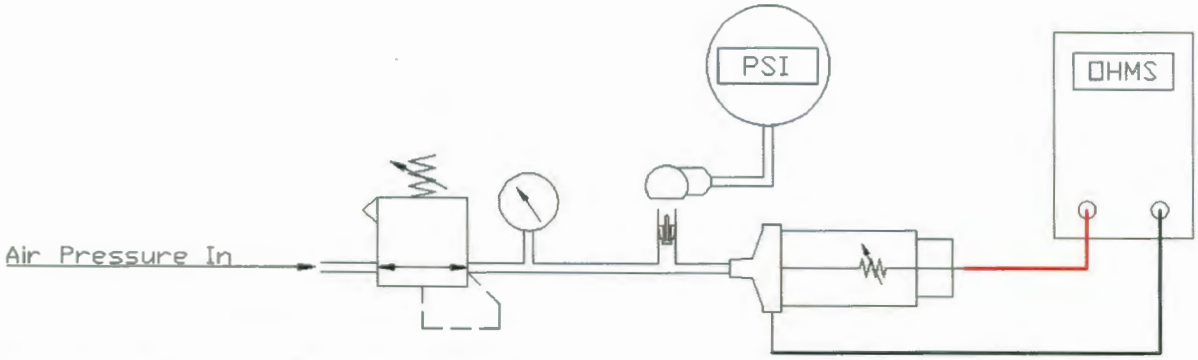


Figure 18. A diagram of the initial pressure sensor test is shown.

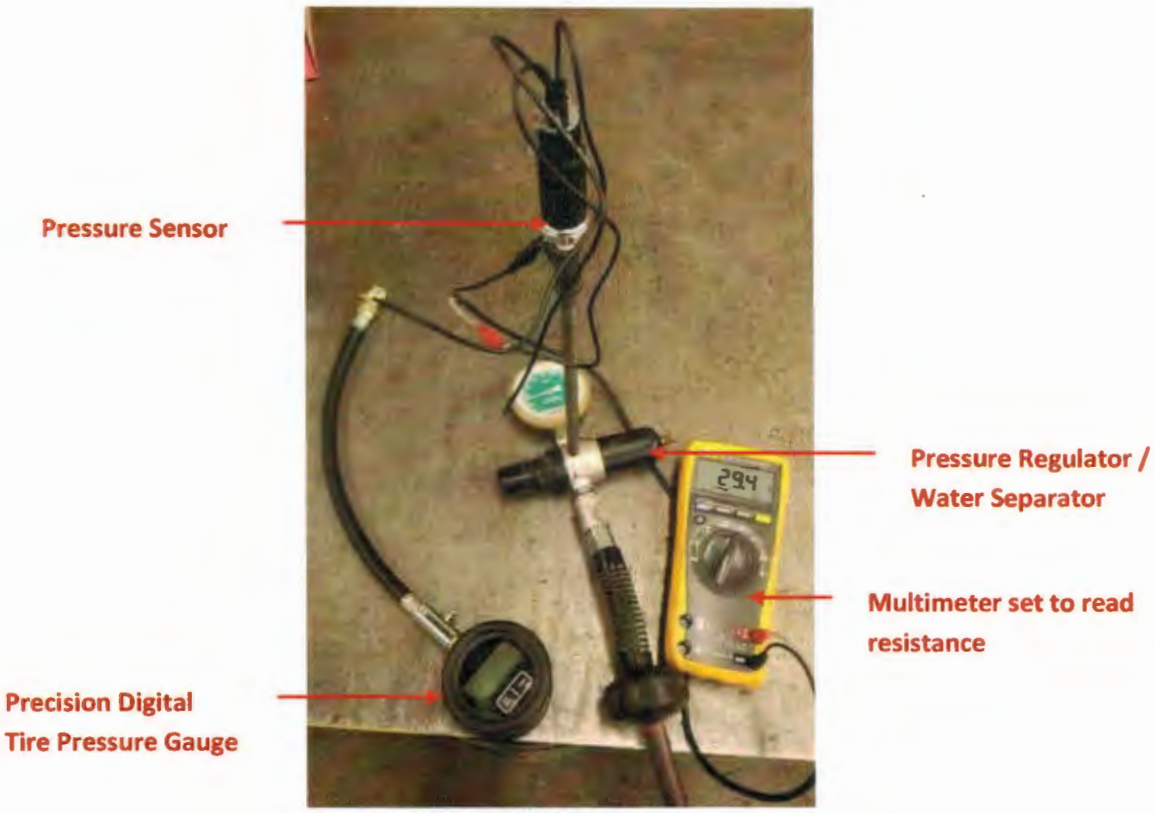


Figure 19. Mapping of the General Motors pressure sensor's response characteristics is shown.

Sensor resistance data was collected from 2.5 to 80 psi in increasing 2.5 psi increments. During this test, as pressure values reached 25, 27.5, and 30 psi, it was noticed that the DMM stopped reading

consistently. It was determined that the pressure sensor internal resistor was most likely a wire-wound configuration, causing some inductance in addition to resistance, thus causing incorrect DMM measurements. At this point the test was halted and the data considered void. Since inductance is a function of conductor length (among other variables), it is suspected it's contribution to impedance was not noticeable until enough resistor wire turns were included.

A modified method of testing sensor response characteristics was developed that provided accurate measurements while utilizing the precision of the Fluke 179 DMM. A high-power adjustable resistor was obtained and calibrated to read 100.0 ohms. This was put in series with the pressure sensor to limit current flow. The DMM was used as an ammeter and a second DMM was added to monitor voltage across a battery source (Figure 20).

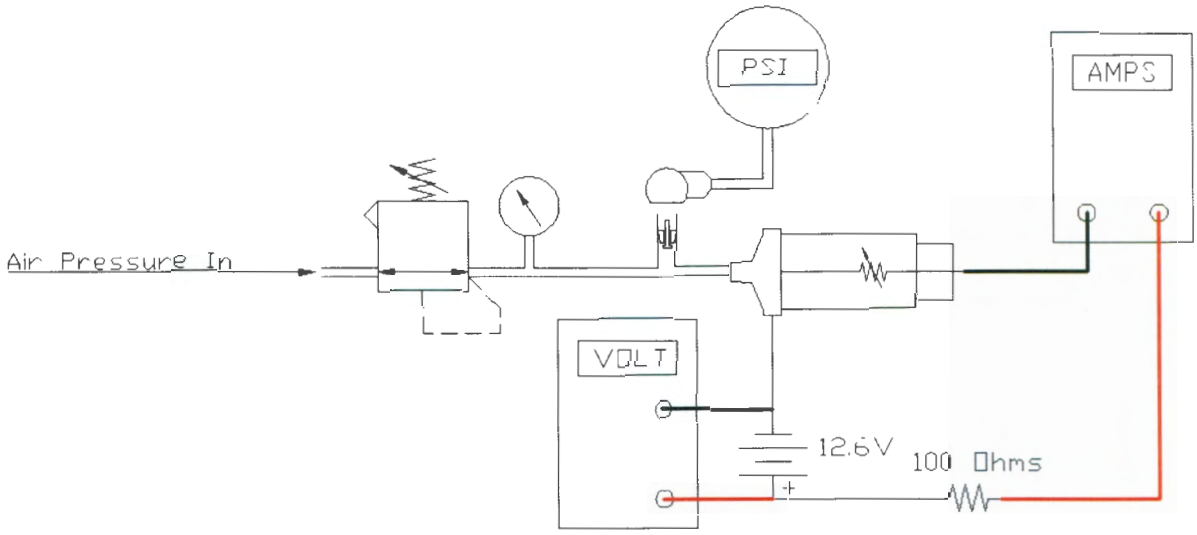


Figure 20. A diagram of the modified pressure sensor test is shown.

Raw data from the second test was entered into a Microsoft Excel spreadsheet, from which a third-order polynomial trend line was determined to best represent its response characteristics (Figure 21). Refer to Appendix G for raw data.

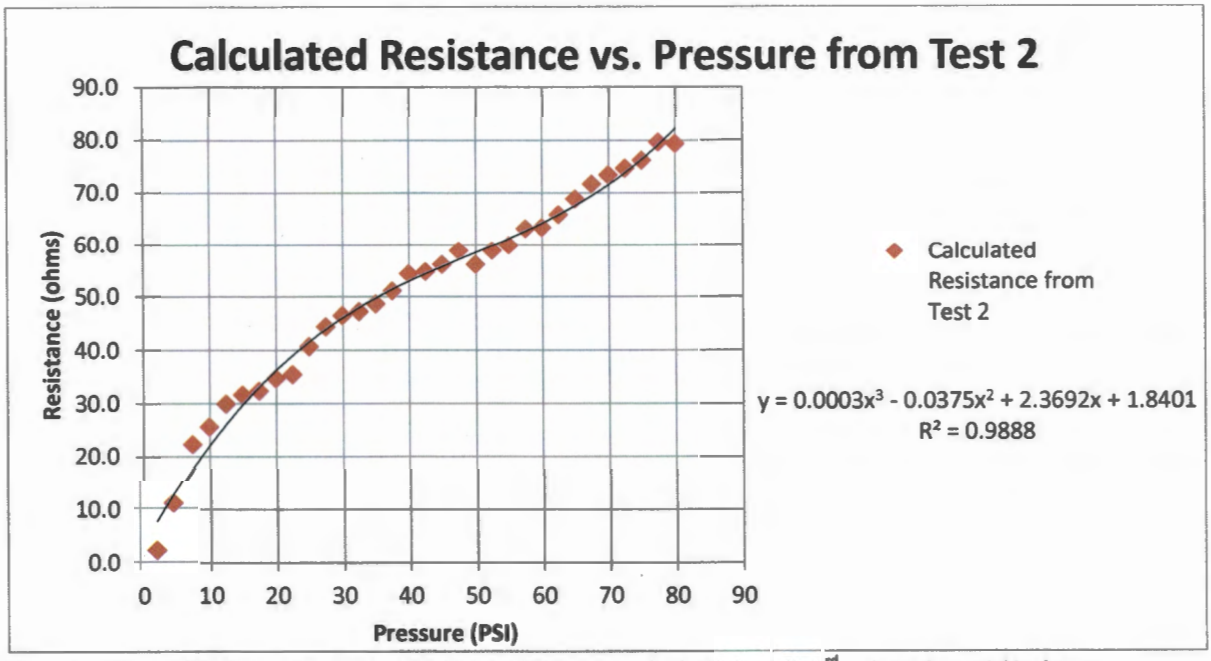


Figure 21. The modified pressure sensor calculated resistances graphed with best-fit 3rd order polynomial is shown.

Oil Pressure Sensor Circuit Design

Based on data from the pressure sensor test, it was determined that a substantial amount of current would be drawn through a voltage-divider circuit in order for the Max32 microcontroller to accurately read the sensor using one of its available analog inputs (Figure 22). Equations 6 and 7 estimate minimum and maximum voltages on the analog input pin. Equation 8 estimates maximum power dissappated in the circuit. Substituting the best-fit polynomial mentioned in Figure 21 and the Max32 assigned value between 0 and 1024, equation 9 relates pressure to analog value.

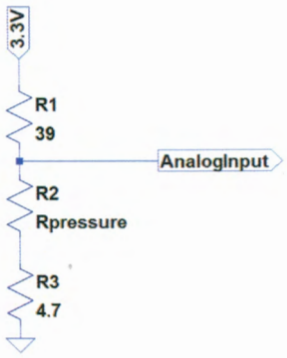


Figure 22. The voltage divider circuit designed for use with the OP6676 sensor (R2) is shown.

$$V_{min} = \left(\frac{0+4.7}{39+0+4.7}\right)3.3V = 0.355V \quad (6)$$

$$V_{max} = \left(\frac{80+4.7}{39+80+4.7}\right)3.3V = 2.260V \quad (7)$$

$$P_{max} = \frac{3.3V}{39+0+4.7}(3.3V) = 0.249W \quad (8)$$

$$Let\ x = pressure(psi),\quad Analog\ Val = (1024)\left(\frac{0.0003x^3-0.0375x^2+2.3692x+6.5401}{0.0003x^3-0.0375x^2+2.3692x+45.5401}\right) \quad (9)$$

Oil Temperature Sensor

Oil temperature is measured using a General Motors OEM-specified engine coolant temperature sensor that varies in resistance with respect to temperature, Echlin part number TS4052. Like the pressure sensor, obtaining this sensor's response characteristics was not easily done so a test platform was also created.

Oil Temperature Sensor Test Procedure:

The temperature sensor was tested using the following components:

1. Aluminum Temperature Sensor Mounting Plate
2. K-type thermocouple with Fluke 179 DMM set in temperature mode
3. A Fluke 8024B DMM set to measure resistance
4. A Royobi HG600 Heat Gun
5. Some aluminum tape

Details of the test setup are shown in Figure 23 and Figure 24.

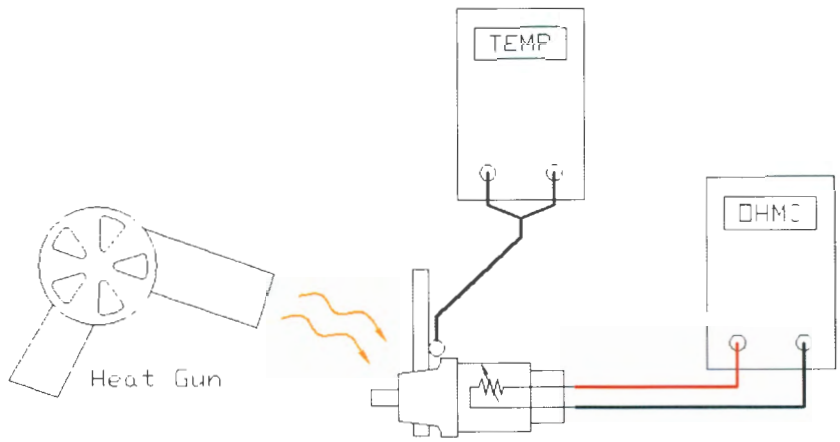


Figure 23. Diagram of the temperature sensor test is shown.

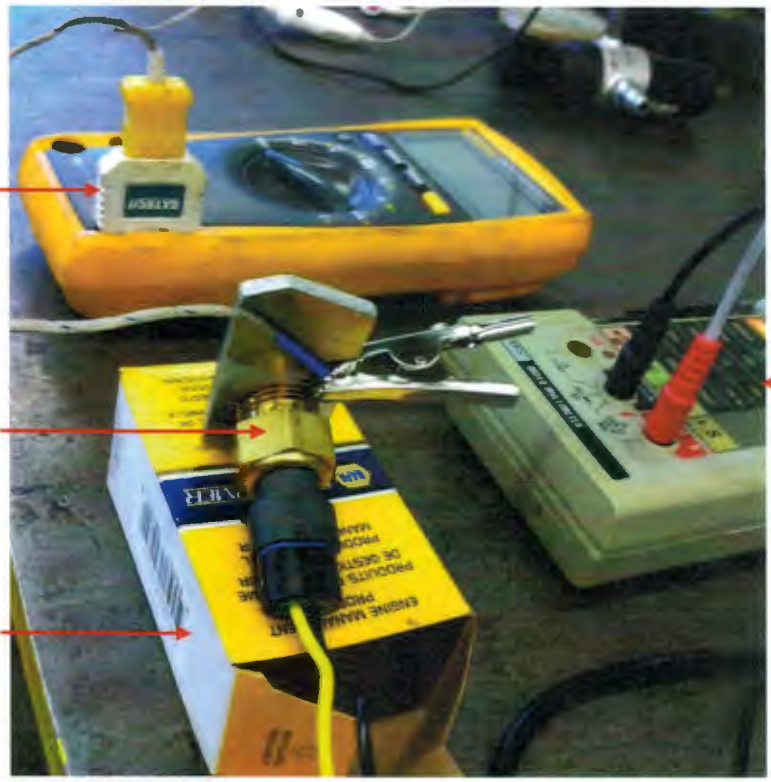


Figure 24. Test setup of the temperature sensor is shown. Note the alligator clip is used to ensure good contact between the aluminum plate and the thermocouple.

The test was conducted by applying the heat gun to the base of the sensor, while monitoring the thermocouple reading on the DMM. Once a "warmer" temperature was reached the heat gun was removed and the sensor, plate, and thermocouple were allowed to cool off. During the cooling process, corresponding temperature and resistance values were recorded in a Microsoft Excel spreadsheet. Though there was no exact process to record specific temperature values (because the sensor was continuously dissipating heat), as many recordings as possible were made, restricted by the swiftness of the human eye. The "heat gun, cool, and record" process was repeated several times until a maximum temperature of 250F was reached, as any higher temperatures could risk damage to the sensor. Refer to Appendix I for raw data from this test. A power-derived trend line, to best represent sensor temperature-resistance response, was created using Microsoft Excel (Figure 25).

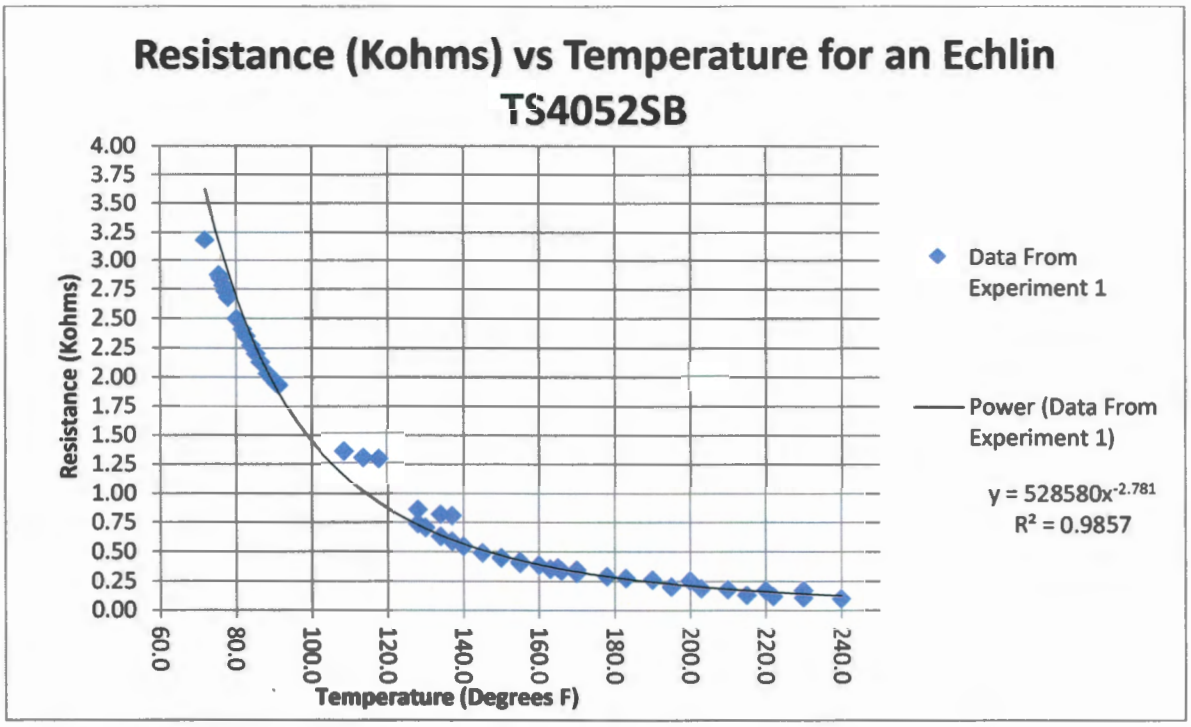


Figure 25. Data graphed and best-fit line from the temperature sensor test is shown.

Temperature Sensor Circuit Design

Like the pressure sensor, a voltage divider circuit sent to one of the microcontroller's analog inputs was used (Figure 26). Equations 10 and 11 estimate minimum and maximum voltages on the analog input pin. Substituting the best-fit polynomial mentioned in Figure 25 and the Max32 assigned value between 0 and 1024, equation 12 relates pressure to analog value.

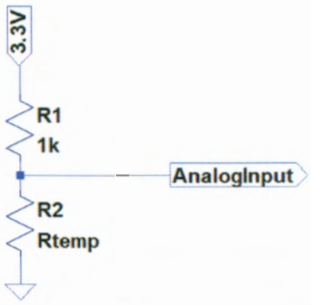


Figure 26. The voltage divider circuit designed for use with the TS4052SB is shown.

$$V_{min} = \left(\frac{100}{100+1000}\right) 3.3V = 0.3V \quad (10)$$

$$V_{max} = \left(\frac{3250}{3250+1000}\right) 3.3V = 2.524V \quad (11)$$

$$Let\ temperature = x,\ Analog\ Val = (1024)\left(\frac{528580x^{-2.781}}{528580x^{-2.781}+1}\right) \quad (12)$$

Integration of Temperature and Pressure Sensor Circuits

Both circuits will not give a linear voltage curve based on their change in resistance due to the nature of using a single voltage divider circuit (as opposed to a Wheatstone Bridge where two sensors are required). The analog inputs utilized for this are scaled in the software accordingly. Since the current draw for the pressure sensor varies substantially, the voltage regulator output driving these two analog outputs was used for the microcontroller's analog reference pin. This allows for any necessary correction in a non-ideal voltage regulator (Figure 27)

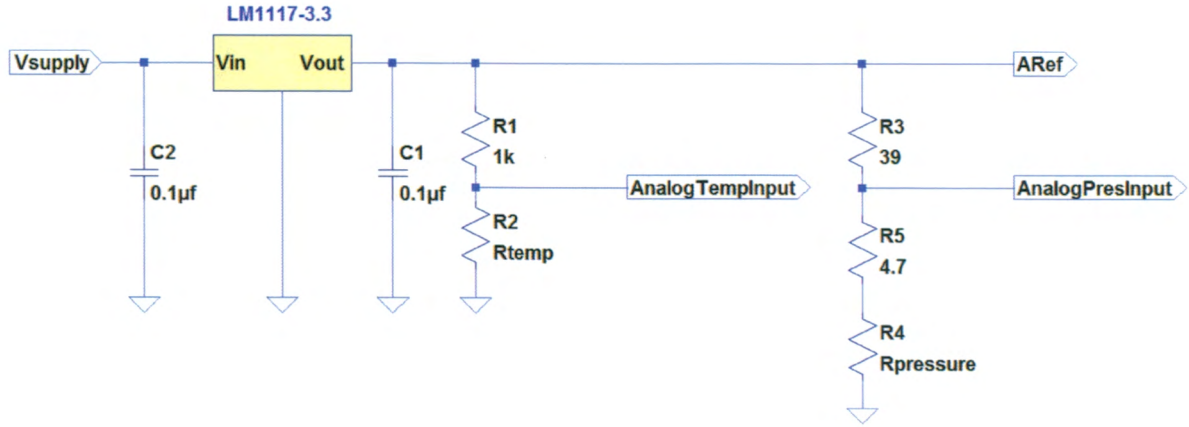
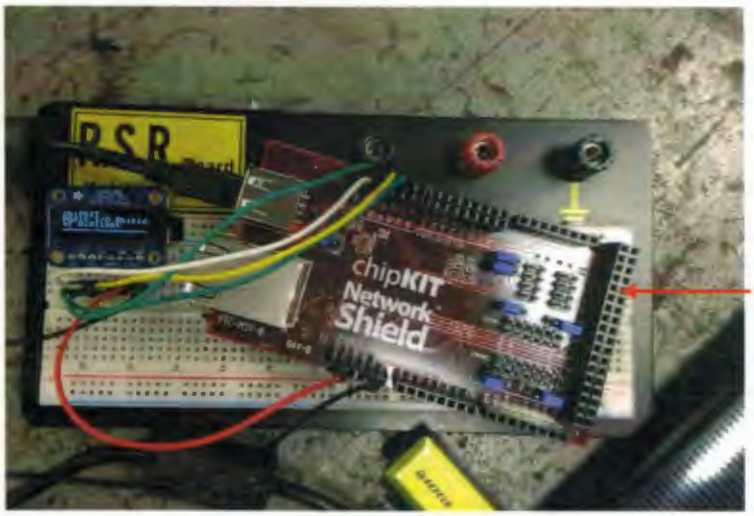


Figure 27. Combined Sensor Circuits are shown. Note that they use a common voltage source, which is also connected to the analog input reference pin (ARef).

Diagnostic Output Screen

A diagnostic screen was added to assist in troubleshooting. Engine speed, along with any analog information received, is displayed on this screen. This information will be available via serial communication, however in the event of troubleshooting where serial communication is not available, this will provide some technician support in pinpointing a circuit that may not be functioning properly.

Due to its low cost, high contrast, and small package, and ease of use, an organic LED display was chosen: model SSD1306 available from Adafruit.com. The SSD1306 is 3.3V compatible, and receives information via SPI connection (serial peripheral interface)- a feature available on the PIC32 processor of the Max32 Microcontroller. The only external components required are a 10K-ohm resistor between the CS (chip select) pin and ground (faulting it to "active"). Since the display will be the only item on this SPI line to the microcontroller, faulting the CS pin to active causes the display to listen for information on the SPI connection at all times (Figure 28).



Max32 Microcontroller with ChipKIT Network Shield attached

Figure 28. Testing of the diagnostic screen is shown.

Another feature unique to the SSD1306 display is its use in another ChipKIT shield – the Digilent ChipKIT Basic I/O Shield (Figure 29). This shield was designed specifically for ChipKIT's Aduino-platform microcontroller line. Due to its use in that shield, SSD1306 libraries specifically compatible with the Max32 are already available, making the integration of this component in the PID control program require little effort. Refer to Appendix I for Basic I/O shield library usage.

SSD1306 OLED Screen

Figure 29. The Digilent / ChipKIT Basic I/O shield is shown (Digilent 2014).

Completed Design

Combining all inputs utilized, along with those necessary for the ChipKIT Network Shield, required close attention to not double-assigning pins. An excel spreadsheet was developed to keep track of every Max32 pin utilized (appendix H). A detailed electrical schematic is shown in Appendix J. Refer to Figure 30 for a complete block diagram of the controller design.

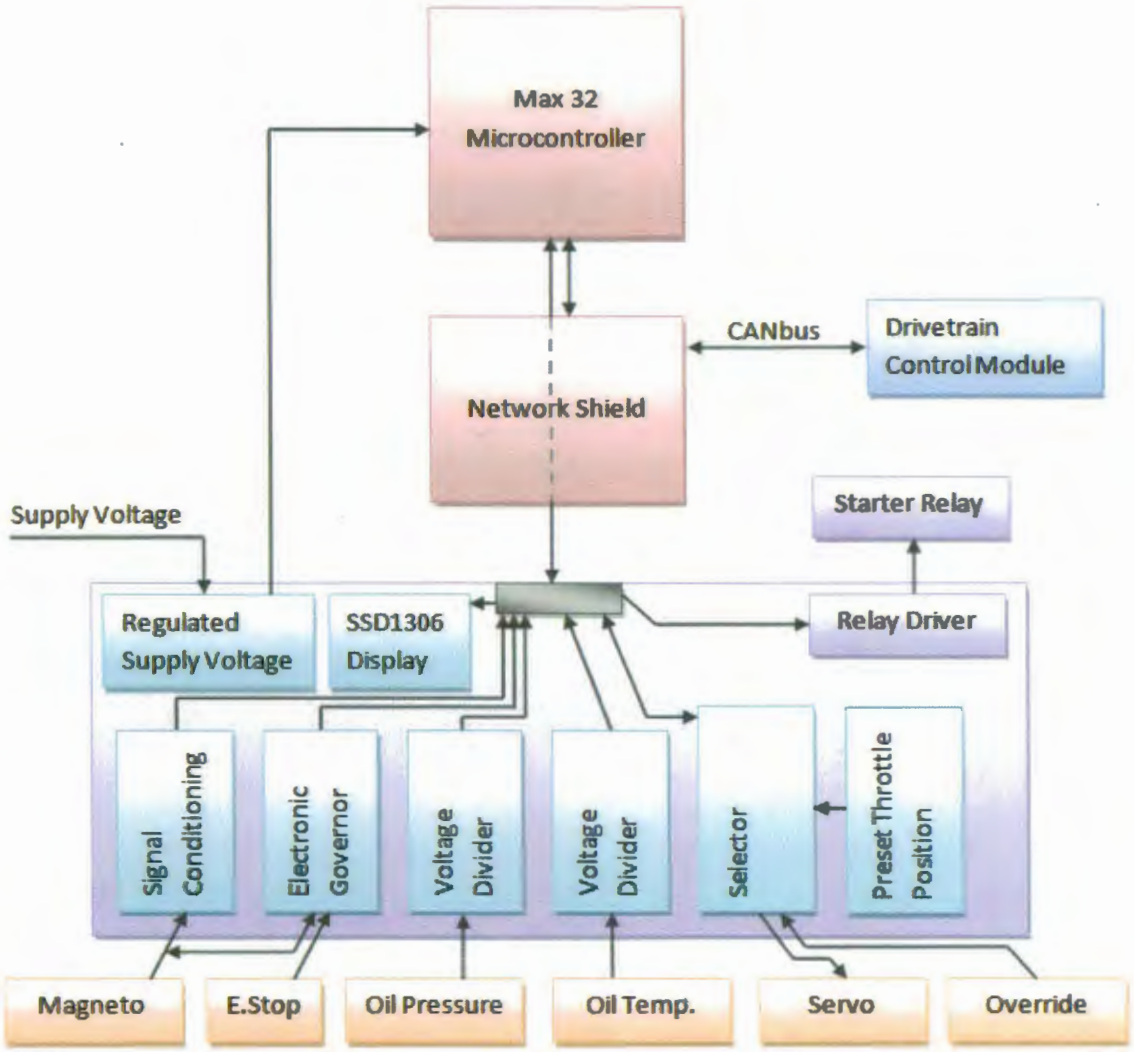


Figure 30. The complete microcontroller block diagram is shown. Note that each arrow may represent multiple transmission lines.

RESULTS

This design and testing project has demonstrated that it is possible to take a two cylinder 4 stroke engine with limited technology: Single-barrel carburetion and fixed-timing magneto ignition, and attach a stand-alone microcontroller. Minimal fabrication is needed, beyond the manufacture of electronics and an appropriate throttle servo mounting plate.

Though possible, the use of a single-pulse-per-revolution engine speed signal proved to have some downfall, being sluggish and difficult to respond to changes at low speeds. The safety circuits added are considered absolutely necessary during the software tuning phase. In a few cases during initial testing of the engine speed signal circuit (refer to *Engine Speed Signal*), the engine throttle did not respond correctly: Engine hesitation (that was not consistent) caused the microcontroller to overshoot the correct throttle position, and severe engine damage was only prevented by quick grounding of the magneto kill wire.

DISCUSSION

The use of the Max32 Microcontroller required several intermediate circuits to adjust signals to be compatible with its input requirements. Though this did add considerable complexity to the project, overall the added components were kept to a minimum. There are several relay driver circuits that could be more efficient if a MOSFET solid-state switching circuit was used as opposed to BJT's.

One concern, yet to be tested, is the microcontroller's durability to vibration. At present the microcontroller is planned to be mounted on top of the engine in an ABS or Polycarbonate enclosure. The solderless headers connecting the microcontroller to the Network Shield and proposed shield would only rely on friction to maintain electrical connections. It is recommended that the enclosure be UV resistant and water tight to prevent corrosion. For development purposes, provided the connections are restrained, the system should be adequate. Should this product go into a commercial-use environment, it would be a better choice to integrate all components onto one multi-layer printed circuit board.

Due to the time constraints of the project, no information was gathered regarding the compatibility of the CANbus serial line available on the Network Shield, and the Danfoss Plus+1 microcontroller.

Testing the controller during engine-loading conditions will also be very important in the next phase of the microcontroller's development. It is suspected that the controller will have better throttle control under constant-load conditions, and see more difficulty in fluctuating engine loading. When under load the manifold will be under less vacuum, and the throttle plate will be further open. Corrections made will span a larger throttle plate angle, and the engine will have more "forgiveness" to over-shooting the correct throttle position.

RECOMMENDATIONS

Final Construction Recommendations

Looking forward, the complete schematic should be re-drawn in EAGLE 6.6.0 (CadSoft, 2014). EAGLE is used to map printed circuit boards for etching. In this software, when components are entered into the schematic, their corresponding package type and size is chosen (SMT vs. thru-hole technology). In the board layout design portion of the program, there is an automatic feature. This however, only has some level of capability and the majority of traces need to be laid out by hand specific to the user's layout requirements (trace size, location, ground planes, etc.)

Using a photo-sensitized copper clad circuit board (Appendix C), manufacturing of the circuit board will involve the following process:

1. Print the completed layout patterns on transparency paper (both top and bottom layers)
2. In very low light, align the top layer to the first side of the sensitized pwb.
3. Expose the pwb to bright red light for 8 minutes.
4. In very low light again, align the bottom layer to the second side of the sensitized pwb
5. Repeat step 3 for the second side.
6. Place pwb in developer solution.
7. Place pwb in etching solution (ferric acid)
8. Place pwb in cleaning solution.
9. Mount plate in vertical mill jig and drill all necessary holes.

Design Improvement Recommendations

In future revisions of this design, it is recommended that engine speed is obtained from an added sensor ring or, if possible, a proximity sensor mounted to count teeth on a crankshaft or camshaft gear. Having sensor pulses between crankshaft revolutions would allow the microcontroller's PID function to make corrections more-quickly, creating possibly a more stable throttle control. Electronically, this improvement can be quickly implemented by changing or possibly even bypassing the filter's 1^{st} clamping stage, provided the pulse signal can at least achieve 3 Volts.

The electronic governor is functional, but not environmentally friendly. Limiting the engine's speed by temporarily cutting the ignition releases large quantities of unburned hydrocarbons into the exhaust. It would be more ideal to have a method of rapidly cutting fuel, or, rapidly overriding the throttle closed (similar to how the original mechanical governor works).

REFERENCES

Bedane, G, M Gupta, D George, and BV. ELSEVEIR SCIENCE. 2008. Development and Evaluation of a Guayule Seed Harvester. *Industrial Crops and Products* 28, no. 2: 177-183.

CadSoft. 2014. EAGLE Light Edition Version 6.6. USA. CadSoft Computer.

Coen, T., W. Saeys, B, Missotten, and J. De Baerdermaeker. 2008. "Cruise control on a combine harvester using model-based predictive control." *Biosystems Engineering* 99, no. 1: 47-55. Academic Search Premier, EBSCOhost (accessed June 12, 2013).

Combs, Charles. 2008. Embedded Java Propulsion Controller for the Cal Poly Rose Float. MS thesis. San Luis Obispo, CA. California Polytechnic State University, Mechanical Engineering Department.

Digilent. 2011. ChipKIT Max32 Board Reference Manual. Pullman, WA.:Digilent, Inc.

Digilent. 2014. ChipKIT Basic I/O Shield. Pullman, WA: Digilent, Inc. Available at: https://digilentinc.com/Products/Detail.cfm?NavPath=2,892,936&Prod=CHIPKIT-BASIC-IO-SHIELD. Accessed 1 June, 2014.

Engelhardt, Mike. 2014. LTspice IV Version 4.20p. Milpitas, CA. Linear Technology Corporation.

Futaba. 2014. Standard Servos. Champaign, IL.: Hobbico, Inc. Available at: http://www.futaba-rc.com/servos/analog.html. Accessed 10 June, 2014.

Kin, Yap Y, Sudhanshu S. Jamuar, and Azmi Yahya. 2011. "COMBINE HARVESTER INSTRUMENTATION SYSTEM FOR USE IN PRECISION AGRICULTURE." *Instrumentation Science & Technology* 39, no. 4:374-393. Academic Search Premier, EBSCOhost (accessed June 12, 2013).

Kohler. 2004. Owner's Manual: Magnum 18 & 20 HP Horizontal Crankshaft. Kohler, Wisconsin. Engine Division, Kohler Co.

Microchip. 2013. 32-Bit Microcontrollers (up to 512KB Flash and 128KB SRAM) with Graphics Interface, USB, CAN, and Ethernet. Chandler, AZ.: Microchip Technology, Inc.

Smith, Carlos A., Armando B. Carripro. 1997. Principles and Practice of Automatic Process Control. 2nd ed. New York, John Wiley & Sons, Inc.

SAE Digital Library. 2013. (J744_201302). Warrendale, PA. Society of Automotive Engineers

Appendix A – Kohler Magnum 18 Specifications

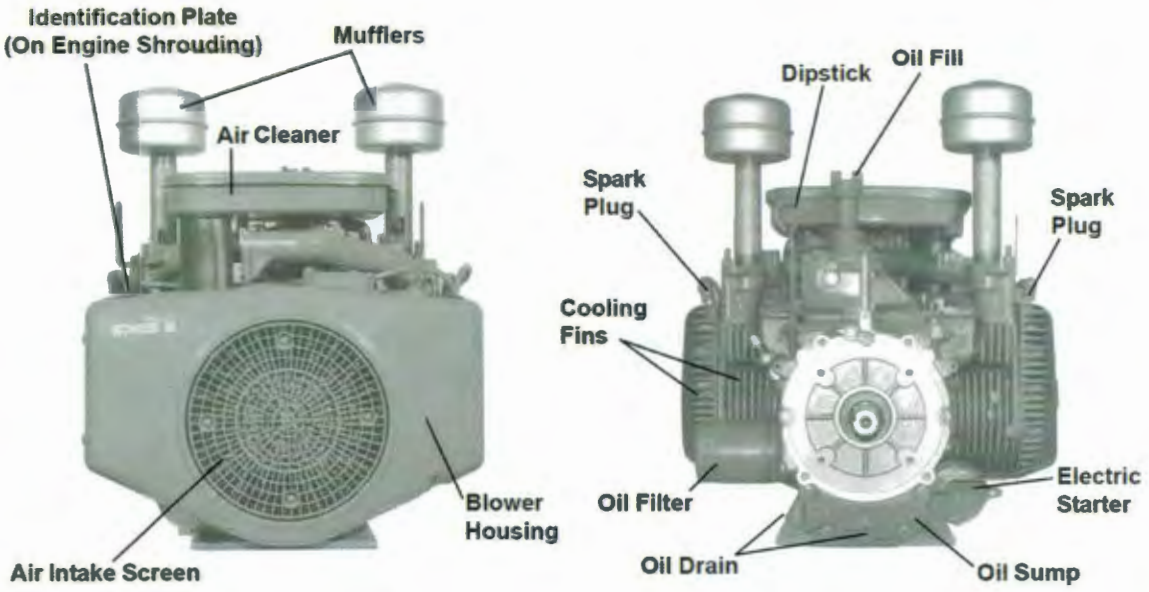


Figure 31. Front and Back views of a typical Kohler Magnum 18 in stock configuration is shown. (Kohler, 2004)

Specifications

Model:		M18	M20
Bore:	inches (millimeters)	3.12 (79.2)	3.12 (79.2)
Stroke:	inches (millimeters)	2.75 (69.85)	3.06 (78.0)
Displacement:	cubic inches (cubic centimeters)	42.18 (691.3)	46.98 (769.8)
Power (@3600 RPM):	horsepower (kilowatts)	18 (13.4)	20 (14.9)
Weight:	lbs. (kilograms)	130 (59.0)	130 (59.0)
Oil Capacity:	U.S. quarts (liters)	1.5 (1.4)[1]	1.5 (1.4)[1]
Spark Plug Gap:	inches (millimeters)	0.035 (0.89)	0.035 (0.89)
Spark Plug Type:	Champion or Equiv.	RV17YC	RV17YC

Figure 32. Kohler Magnum M18 and M20 Mechanical Specifications are shown. (Kohler, 2004)

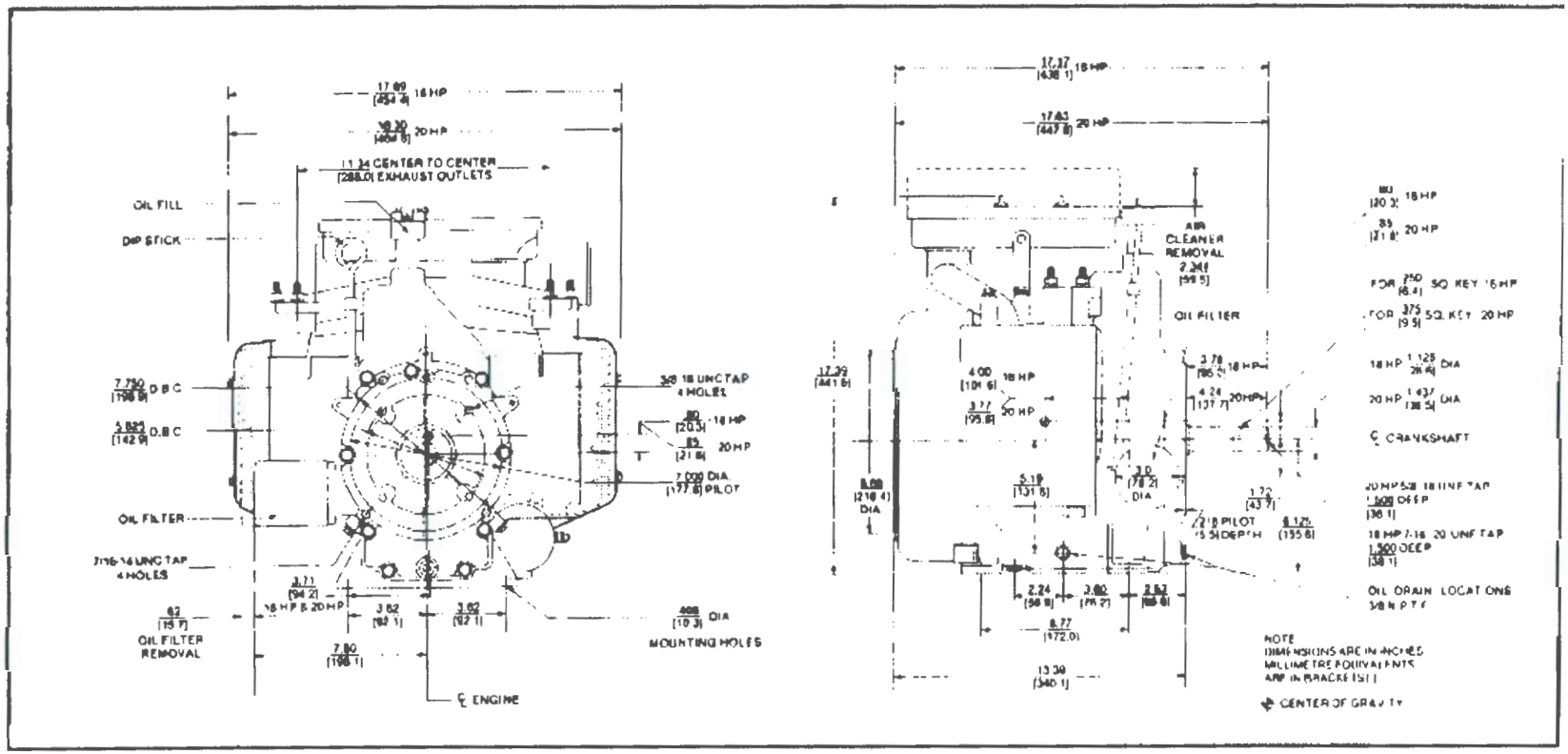


Figure 33. The available drawing of Kohler Magnum 18 external dimensions is shown (Kohler, 2004).

Appendix B – SAE-A 9T 16/32 Specifications

Table 1. SAE A dimensions in inches in reference to Figure 34 (SAE, 2013).

Identification Code	Pilot Dimension A	Pilot Dimension W	Pilot Dimension X	Pilot Dimension Y	2 Bolt Type B Cast Dim	2 Bolt Type J	2 Bolt Type K	2 Bolt Type M	2 Bolt Type P1 Cast Dim
A	3.25	0.25	-	0.03	3.75	0.72	4.188	0.438	0.47

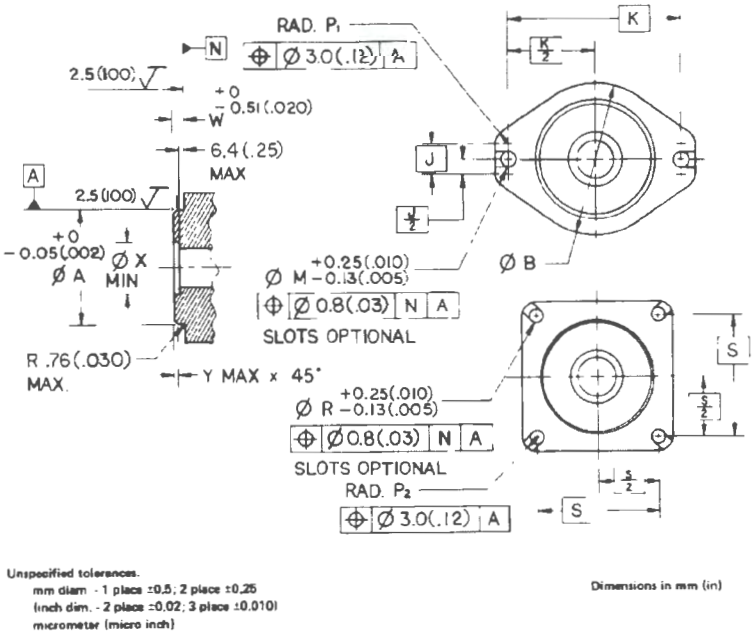


Figure 34. SAE Hydraulic Pump and Motor Mounting Dimensions are shown (SAE, 2013).

Table 2. Dimensions of 30 degrees involute spline shafts in inches in reference to Figure 35 is shown

Identification Code	Spline	U Min	L_A Min	L_{ss}	L_B
16-4(A)	9T 16/32 DP	0.4650	0.30	0.938	0.06

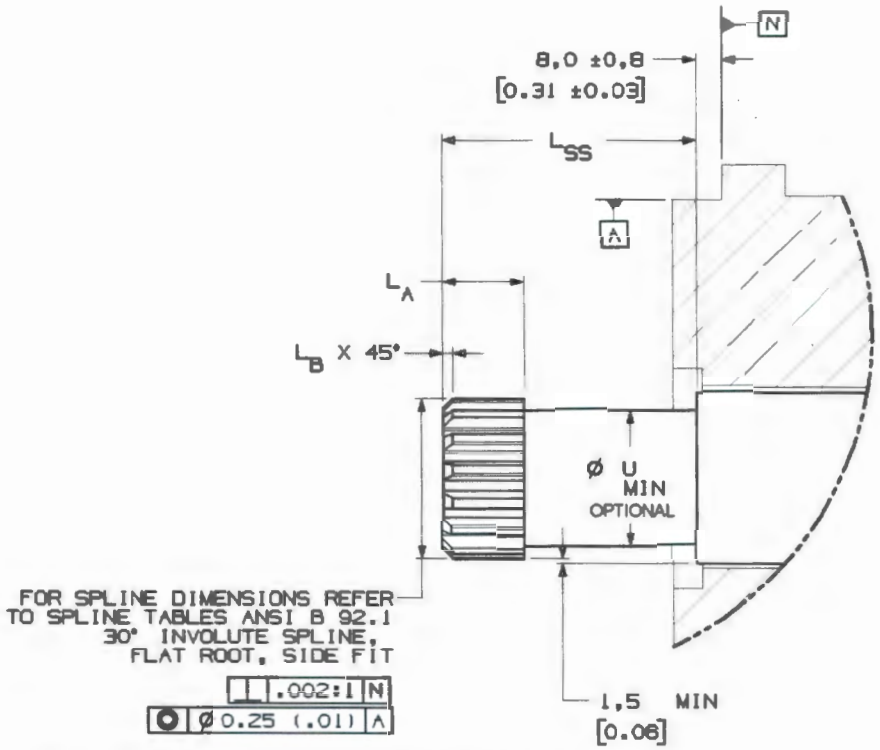


Figure 35. SAE Hydraulic Pump and Motor spline shaft drive dimensions are shown (SAE, 2013).

Appendix C – Sourced Components

Table 3. A table of select components purchased for the throttle controller is shown.

Where Used	Brand - Description	Part Number	Approximate Cost
Linkage Turnbuckle	Team Losi – Front/Rear Steering Turnbuckle Set (6) 10-T	LOSB4003	$13/set
Linkage Rod Ends	DU-BRO Racing - 4-40 E/Z Adjust Ball Links (Short) With Hardware	2141	$3/pr
Actuator	Futaba High Torque Standard Servo	S3305	$35/ea
Circuit Board	DATAK – Double Sided 3.0" x 4.0" Pre-Sensitized Circuit Board	14-234	$6/ea
Oil Pressure Sensor	Echlin – OEM Replacment Oil Pressure Sender	OP6676	$27/ea
Oil Pressure Sensor Connector	Echlin – OEM Replacment Electrical Connector	EC78	$21/ea
Oil Temperature Sensor	Echlin – OEM Replacment Coolant Temperature Sensor	TS4052SB	$20/ea
Oil Temperature Sensor Connector	Echlin – OEM Replacment Electrical Connector	TSC200SB	$14/ea

Appendix D – Futaba S3305 Specifications

Table 4. Futaba S3305 Specifications are shown (Futaba, 2014).

Volts	Torque	Speed
4.8V	99 oz-in	0.25 sec/60 degrees
6.0V	124 oz-in	0.20 sec/60 degrees

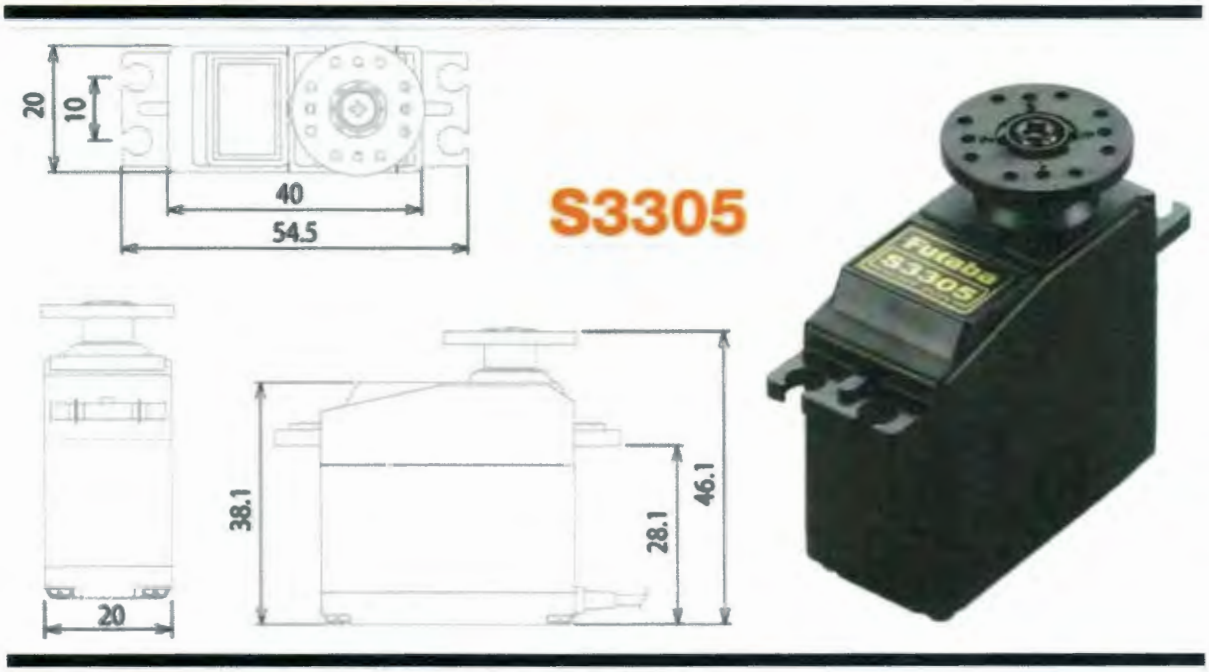


Figure 36. Futaba S3305 Dimensions are shown (Futaba, 2014).

Appendix E – ChipKIT Digilent Max32 Specifications

Table 5. Digilent ChipKIT Max32 Microcontroller Specifications are shown (Digilent, 2011).

Microcontroller	PIC32MX795F512L
Flash Memory	512K
RAM Memory	128K
Operating Voltage	3.3V
Operating Frequency	80MHz
Typical operating current	90mA
Input Voltage (recommended)	7V to 15V
Input Voltage (maximum)	20V
I/O Pins	83 Total
Analog Inputs	16
Analog input voltage range	0V to 3.3V
DC Current per pin	+/-18mA

Table 6. PIC32 Digital Input Comparator Reference Voltage Characteristics (Microchip, 2013).

Symbol	Characteristics	Min.	Typical	Max	Units
IV_{REF}	Internal Voltage Reference	0.57	0.6	0.63	Volts
V_{IOFF}	Input Offset Voltage	-	7.5	25	mV

Appendix F – Microcontroller Program Source Code

Program Open source Code is as follows:

```
#define STALL_TIMEOUT 125000
#define STARTSOLTIME 3000

/*
Pin definitions
*/
#define STARTSOLPIN 4

/*
TACHTIMECONSTANT = 4rev * 1000000 us/s * 60 sec/min
*/
#define TACHTIMECONSTANT 240000000

volatile unsigned long engTimeFour = 0;
volatile unsigned long engTimeThree = 0;
volatile unsigned long engTimeTwo = 0;
volatile unsigned long engTimeOne = 0;
volatile unsigned long engTimeZero = 0;

int startSol = 0;
int startSolPinStat = LOW;
String message = "#1000F0";



/*
What the program does during external interrupt 0:
Shifts time values up one level
*/

void onTachPulseInterrupt(void) {
  engTimeFour = engTimeThree;
  engTimeThree = engTimeTwo;
  engTimeTwo = engTimeOne;
  engTimeOne = engTimeZero;
  engTimeZero = micros();
}

/*
What the program does during external interrupt 1:
Writes 1 to crank variable
*/
```

```
void crankEngOverride(void) (
 startSol = 1;
}

/*
Fuction definition of rpm() will return value of rpm
*/

int rpm() {
  unsigned long local_oldtime;
  unsigned long local_curtime;

  noInterrupts();
  local_oldtime = engTimeFour;
  local_curtime = engTimeZero;
  interrupts();

  if ((micros() - local_curtime) < STALL_TIMEOUT) {
    return TACHTIMECONSTANT / (local_curtime - local_oldtime);
  }
  else {
    return 0;
  }
 }

int startSeq() {
  unsigned long startSolTimer;

  if (rpm() > 0) {
   startSolPinStat = LOW;
   startSol = 0;
  }
  else {
   if (startSolPinStat == LOW) {
    startSolTimer = millis();
   }
   if (startSolTimer + STARTSOLTIME > millis() && startSolPinStat == LOW ) {
    startSolPinStat = HIGH;
   }
   else {
    startSolPinStat = LOW;
    startSol = 0;
   }
  }
  digitalWrite(STARTSOLPIN, startSolPinStat);
```

```
}


void setup () {
 attachInterrupt(0, onTachPulseInterrupt, FALLING);
 attachInterrupt(1, crankEngOverride, FALLING);

 pinMode(STARTSOLPIN, OUTPUT);
 digitalWrite(STARTSOLPIN, LOW);
 Serial.begin(9600);
 Serial1.begin(9600);
}

void loop () {
 if (startSol == 1) {
  startSeq();

 }

 String serialStringIn = "";
 if (Serial.available() > 0) {
  while (Serial.available() > 0) {
   serialStringIn += char(Serial.read());
   delay(250);
  }
 }

 Serial1.println(serialStringIn);
 String serialString = "";
 if (Serial1.available() > 0) {
  while (Serial1.available() > 0) {
   serialString += char(Serial1.read());
   delay(250);
   Serial.println(serialString);
  }
 }

}
```

Appendix G – Sensor Testing Raw Data

Oil Pressure Sensor Test 2

Table 7. Raw data and calculated resistance pressure sensor resistance values are shown.

Pressure	Battery Voltage	Load Resistor	Current Measured	Calculated Sensor Resistance
2.5	12.08	100.0	118.3	2.1
4.9	12.08	100.0	108.9	10.9
7.5	12.08	100.0	98.8	22.3
10	12.08	100.0	96.2	25.6
12.5	12.08	100.0	93.0	29.9
15	12.07	100.0	91.7	31.6
17.5	12.07	100.0	91.2	32.3
20	12.07	100.0	89.6	34.7
22.5	12.07	100.0	89.1	35.5
25	12.07	100.0	85.8	40.7
27.5	12.07	100.0	83.6	44.4
30	12.07	100.0	82.4	46.5
32.5	12.07	100.0	82.0	47.2
35	12.07	100.0	81.2	48.6
37.5	12.07	100.0	79.9	51.1
40	12.07	100.0	78.2	54.3
42.5	12.07	100.0	78.0	54.7
45	12.07	100.0	77.3	56.1
47.5	12.07	100.0	76.0	58.8
50	12.07	100.0	77.3	56.1
52.5	12.07	100.0	76.0	58.8
55	12.07	100.0	75.5	59.9
57.5	12.07	100.0	74.0	63.1
60	12.07	100.0	73.9	63.3
62.5	12.07	100.0	72.8	65.8
65	12.07	100.0	71.5	68.8
67.5	12.07	100.0	70.3	71.7
70	12.07	100.0	69.6	73.4
72.5	12.07	100.0	69.1	74.7
75	12.07	100.0	68.5	76.2
77.5	12.07	100.0	67.2	79.6
80	12.07	100.0	67.3	79.3

Oil Temperature Sensor

Table 8. Raw data from the temperature sensor test is shown.

Temp (F)	Resistance (kohms)	Temp (F)	Resistance (kohms
71.9	3.18	145.0	0.492
91.3	1.93	140.0	0.548
90.5	1.95	137.0	0.59
89.5	1.99	134.0	0.632
88.4	2.03	130.0	0.706
86.5	2.13	128.0	0.739
85.3	2.2	200.0	0.257
84.0	2.27	190.0	0.264
82.7	2.35	183.0	0.278
81.7	2.41	178.0	0.293
80.2	2.5	170.0	0.322
77.9	2.68	166.0	0.34
77.2	2.73	163.0	0.356
76.7	2.78	155.0	0.405
75.9	2.84	230.0	0.175
75.5	2.87	220.0	0.175
118.0	1.28	210.0	0.182
114.0	1.29	203.0	0.191
109.0	1.343	195.0	0.205
137.0	0.808	240.0	0.1047
134.0	0.82	230.0	0.1135
128.0	0.86	222.0	0.1235
170.0	0.348	215.0	0.1339
165.0	0.367		
160.0	0.39		
155.0	0.419		
150.0	0.45		

Appendix H – Microcontroller Pin Assignments

Table 9. Pin assignments for the Max32 microcontroller are shown.

chip KIT Pin #	Connector Pin #	PIC32 Pin	PIC32 Signal	Notes	Used By
0	J14-01	52	SDA1A/SDI1A/U1ARX/RF2		
1	J14-03	53	SCL1A/SDO1A/U1ATX/RF8		
2	J14-05	18	AERXD0/INT1/RE8		USB Interface
3	J14-07	72	SDO1/OC1/INT0/RD0		
4	J14-09	74	SOSCO/T1CK/CN0/RC14		
5	J14-11	76	OC2/RD1		
6	J14-13	77	OC3/RD2		
7	J14-15	19	AERXD1/INT2/RE9		Ethernet Interface
8	J3-01	79	ETXD2/IC5/PMD12/RD12		
9	J3-03	78	OC4/RD3		ServoSignalOut
10	J3-05	81	OC5/PMWR/CN13/RD4		SSD1306 RST
11	J3-07	9	T5CK/SDI1/RC4		SSD1306 DATA
12	J3-09	58	SCL2/RA2		I2C Interface
13	J3-11	59	SDA2/RA3		SSD1306 CLK
14	J4-08	39	AC1TX/SCK3A/U3BTX/U3ARTS/RF13		CAN Interface
15	J4-07	40	AC1RX/SS3A/U3BRX/U3ACTS/RF12		CAN Interface
16	J4-06	50	SCL3A/SDO3A/U3ATX/PMA8/CN18/RF5		
17	J4-05	49	SDA3A/SDI3A/U3ARX/PMA9/CN17/RF4		
18	J4-04	48	AETXD1/SCK1A/U1BTX/U1ARTS/CN21/RD15		
19	J4-03	47	AETXD0/SS1A/U1BRX/U1ACTS/CN20/RD14		
20	J4-02	67	AETXEN/SDA1/INT4/RA15		I2C Interface
21	J4-01	66	AETXCLK/SCL1/INT3/RA14		I2C Interface
22	J9-16	7	T3CK/AC2TX/RC2		CAN Interface
23	J9-15	8	T4CK/AC2RX/RC3		CAN Interface
24	J9-14	54	VBUS		
25	J9-13	51	USBID/RF3		USB Interface
26	J9-12	56	D-/RG3		USB Interface
27	J9-11	57	D+/RG2		USB Interface
28	J9-10	1	AERXERR/RG15		ECUMAIN
29	J9-09	11	ECRX/SDA2/SDI2A/U2ARX/PMA4/CN9/RG7		StartSol
30	J9-08	5	PMD7/RE7		StartPin
31	J9-07	4	PMD6/RE6		TachPulse
32	J9-06	3	PMD5/RE5		OverRunFault
33	J9-05	100	PMD4/RE4		ServoOverrideStatus
34	J9-04	99	PMD3/RE3		ServoOverridePin
35	J9-03	98	PMD2/RE2		highLimitPin

36	J9-02	94	PMD1/RE1		lowLimitPin
37	J9-01	93	PMD0/RE0		
38	J8-16	70	SCK1/IC3/PMCS2/PMA15/RD10		
39	J8-15	82	PMRD/CN14/RD5		
40	J8-14	35	AN11/EREXERR/AETXERR/PMA12/RB11	Also J7-04(65)	Ethernet Interface
41	J8-13	42	AN13/ERXD1/AECOL/PMA10/RB13	Also J7-06(66)	Ethernet Interface
42	J8-12	41	AN12/ERXD0/AECRS/PMA11/RB12	Also J7-05(67)	Ethernet Interface
43	J8-11	12	ERXDV/AERXDV/ECRSDV/AECRSDV/SCL2A/SDO2A/U2ATX/	Also JP3,JP4	Ethernet Interface
44	J8-10	29	VREF+/CVREF+/AERXD3/PMA6/RA10	also J3-15	
45	J8-09	87	C1RX/ETXD1/PMD11/RF0		Ethernet Interface
46	J8-08	88	C1TX/ETXD0/RMD10/RF1		Ethernet Interface
47	J8-07	83	ETXEN/PMD14/CN15/RD6		Ethernet Interface
48	J8-06	68	RTCC/EMDIO/AEMDIO/IC1/RD8		Ethernet Interface
49	J8-05	71	EMDC/AEMDC/IC4/PMCS1/PMA14/RD11		Ethernet Interface
50	J8-04	11	ECRX/SDA2/SDI2A/U2ARX/PMA4/CN9/RG7	Also JP3,JP4	
51	J8-03	12	ERXDV/AERXDV/ECRSDV/AECRSDV/SCL2A/SDO2A/U2ATX	Also JP3,JP4	
52	J8-02	10	ECOL/SCK2A/U2BTX/U2ARTS/PMA5/CN8/RG6	Also J13-03	
53	J8-01	14	ERXCLK/AERXCLK/EREFCLK/AEREFCLK/SS2A/U2BRX/U2ACTS	Also J13-05	Ethernet Interface
54	J5-01	25	PGED1/AN0/CN2/RB0	Also A0	POTPin
55	J5-02	24	PGEC1/AN1/CN3/RB1	Also A1	Oil Pressure
56	J5-03	23	AN2/C2IN-/CN4/RB2	Also A2	Oil Temp
57	J5-04	22	AN3/C2IN+/CN5/RB3	Also A3	EGT
58	J5-05	21	AN4/C1IN-/CN6/RB4	Also A4	
59	J5-06	20	AN5/C1IN+/VBUSON/CN7/RB5	Also A5	USB Interface
60	J5-7	26	PGEC2/AN6/OCFA/RB6	Also A6	
61	J5-8	27	PGED2/AN7/RB7	Also A7	
62	J7-01	32	AN8/C1OUT/RB8	Also A8	
63	J7-02	33	AN9/C2OUT/RB9	Also A9	
64	J7-03	34	AN10/CVREFOUT/PMA13/RB10	Also A10	
65	J7-04	35	AN11/EREXERR/AETXERR/PMA12/RB11	Also A11,J8-14(40)	
66	J7-06	42	AN13/ERXD1/AECOL/PMA10/RB13	Also A12,J8-13(41)	
67	J7-05	41	AN12/ERXD0/AECRS/PMA11/RB12	Also A13,J8-12(42)	
68	J7-07	43	AN14/ERXD2/AETXD3/PMALH/PMA1/RB14	Also A14	
69	J7-08	44	AN15/ERXD3/AETXD2/OCFB/PMALL/PMA0/CN12/RB15	Also A15	
70	J14-02	17	TMS/RA0		
71	J14-04	38	TCK/RA1		
72	J14-06	60	TDI/RA4		
73	J14-08	61	TDO/RA5		
74	J14-10	69	SS1/IC2/RD9		

75	J14-12	73	SOSCI/CN1/RC13		
76	J14-14	80	ETXD3/PMD13/CN19/RD13		
77	J14-16	84	ETXCLK/PMD15/CN16/RD7		
78	J3-02	89	C2TX/ETXERR/PMD9/RG1		
79	J3-04	90	C2RX/PMD8/RG0		
80	J3-06	91	TRCLK/RA6		
81	J3-08	92	TRD3/RA7		
82	J3-10	95	TRD2/RG14		
83	J3-12	96	TRD1/RG12		SSD1306 D/C
84	J3-14	97	TRD0/RG13		
85	J3-16	28	VREF-/CVREF0/AERXD2/PMA7/RA9		
Power Connector				Notes	Used By
0	J2-01		RST		
1	J2-02		3V3		ECU3.3V
2	J2-03		5V0		ECU5V
3	J2-04		GND		GND
4	J2-05		GND		GND
5	J2-06		VIN		ECUPWR

Appendix I – ChipKIT Digilent Basic I/O Shield SSD1306 Usage

Table 10. Pins relevant to the Basic I/O Shield library for use with the SSD1306 are shown (Digilent, 2014).

#define (I/O library)	ChipKIT Pin #	PIC32 Signal	SSD1306 Usage
bitMosi	11	T5CK/SDI1/RC4	DATA
bitMiso	12	SCL2/RA2	-
bitSck	13	SDA2/RA3	CLK
bitVddCtrl	82	TRD2/RG14	-
bitVbatCtrl	84	TRD0/RG13	-
bitDataCmd	83	TRD1/RG12	D/C
bitReset	10	OC5/PWMR/CN13/RD4	RST

Appendix J – Controller Complete Electrical Schematic

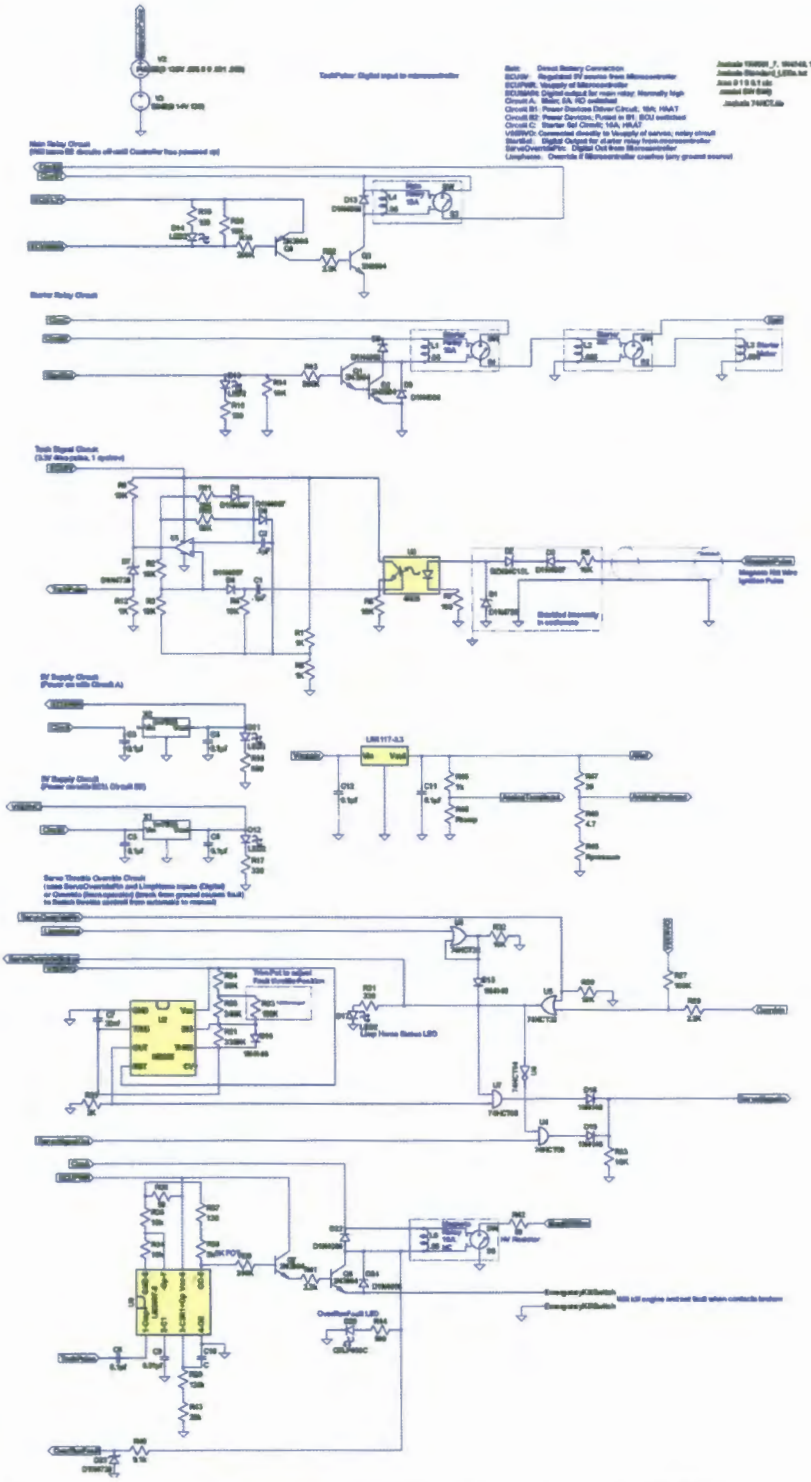


Figure 37. A complete schematic of the necessary components to interface to the microcontroller is shown.