Engine Governor Control System

Duke University, Fall 2015, ECE 495
Authored by Zachary Podbela & Pranava Raparla

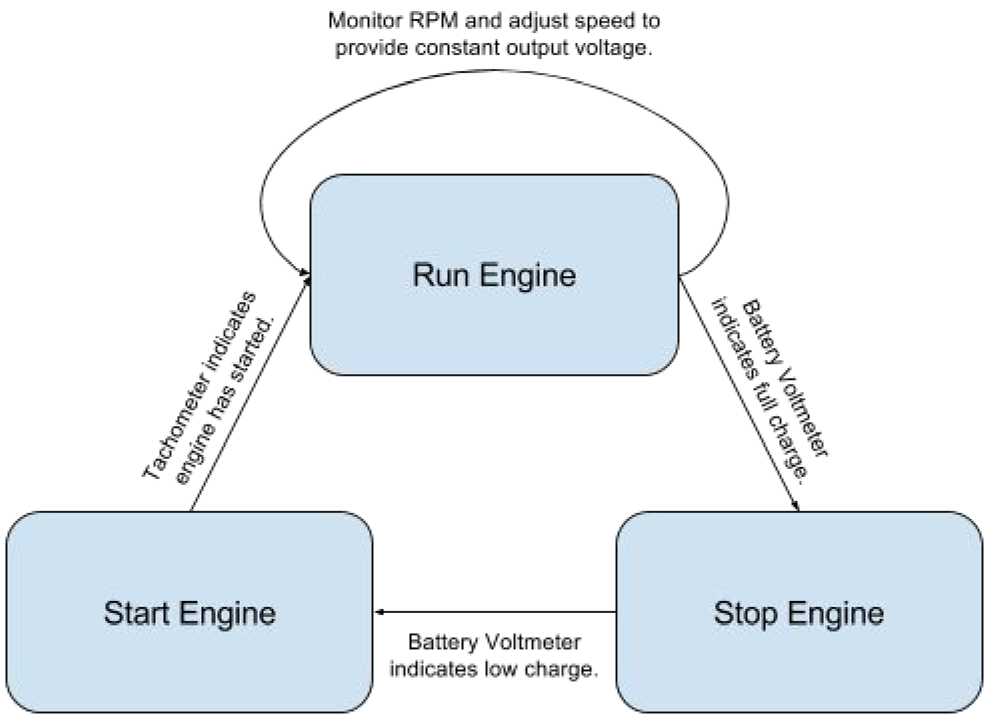
Overview

Our group was tasked with designing & implementing an autonomous system to control the three gas generator engines that will be mounted on the copter.

Requirements

- Coordinate control of an engine mounted starter motor, throttle servo, choke servo, and ignition sparking mechanism.
- System must move between three states—a sequence for starting the engine, managing speed during operation, and stopping engine when prompted.
- Utilize battery voltmeter and engine tachometer readings to move between these three states (See Operation Lifecycle)
- Engine operations must be controlled autonomously by an Arduino.

Operation Lifecycle (Autonomous System State Diagram)


Monitor RPM and adjust speed to provide constant output voltage.
Run Engine
Tachometer indicates engine has started.
Battery Voltmeter indicates full charge.
Start Engine
Stop Engine
Battery Voltmeter indicates low charge.

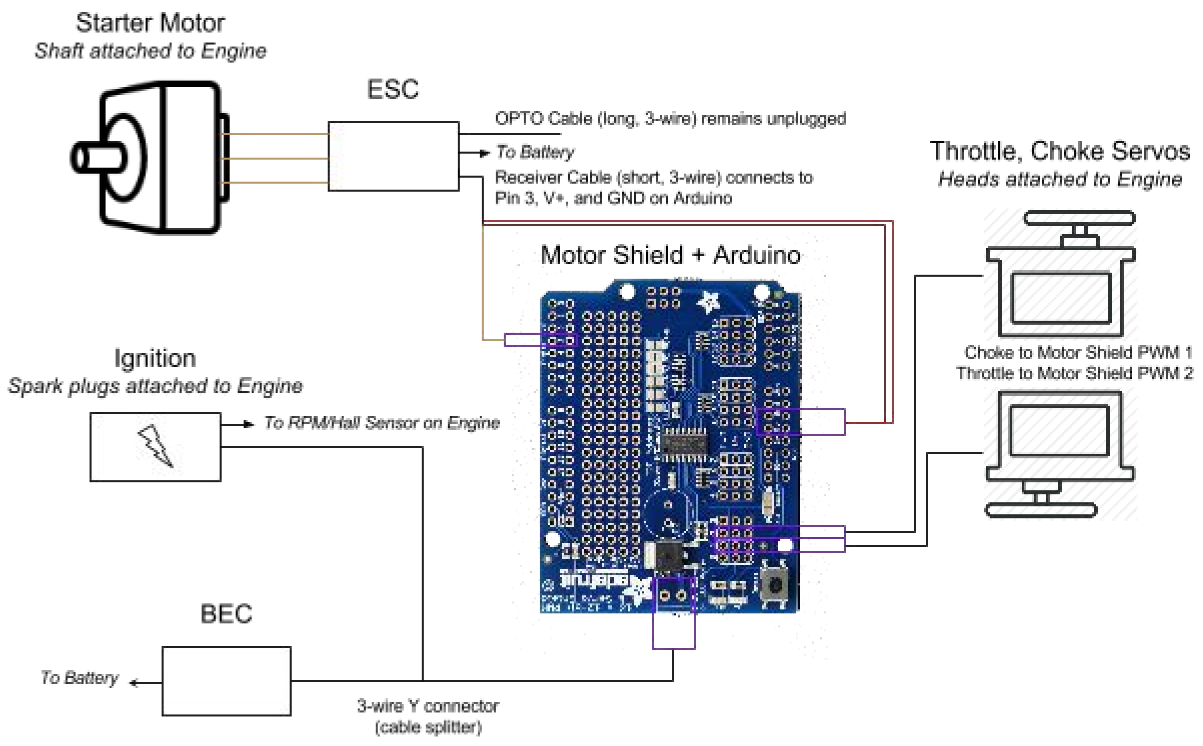
Implementation

Over the course of the semester we built a prototype implementation of this system that will provide a solid foundation for the production iteration. The details of the system including parts list, component wiring diagram, digital resources, assembly instructions, and comments are included below.

Parts List

(1)	IEIK Uno R3
(1)	Adafruit 16-Channel 12-bit PWM/Servo Shield - I2C interface
(2)	TSX35 Standard Servo
(1)	Turnigy RotoMax 80cc Size Brushless Outrunner Motor
(1)	RotorStar 120A HV (4~14S) Brushless Speed Controller
(1)	Rcexl Automatic Ignition CDI
(1)	BEC 3A for 6S LiPo Batteries
(1)	Turnigy nano-tech 5000mah 6S 45~90C Lipo Pack

Component Wiring Diagram


Starter Motor
Shaft attached to Engine
ESC
OPTO Cable (long, 3-wire) remains unplugged
To Battery
Receiver Cable (short, 3-wire) connects to Pin 3, V+, and GND on Arduino
Throttle, Choke Servos
Heads attached to Engine
Motor Shield + Arduino
Choke to Motor Shield PWM 1
Throttle to Motor Shield PWM 2
Ignition
Spark plugs attached to Engine
To RPM/Hall Sensor on Engine
BEC
To Battery
3-wire Y connector
(cable splitter)

Included Resources

All Arduino code is provided in our gitlab repository[1]. The top level of the repository includes the following items:

↳ Adafruit-PWM-Servo-Driver-Library-master.zip   Zip Archive File

> Arduino library provided by Adafruit. Required to use the Adafruit 16-Channel 12-bit PWM/Servo Shield - I2C interface.

↳ generator_controller   Arduino Project Directory

> This Arduino project contains the latest program for running the generator and sending commands to move it through its life cycle. Upload this program to the Arduino controlling your system. This was the main deliverable of the project.

↳ rc_mimic_utility   Arduino Project Directory

> This Arduino project contains a program for mimicking the output of a RC transmitter and receiver setup. It is useful for programming the ESC, which has menus and setup procedures that are navigated by through the movement of an RC throttle.

↳ servo_test_utility   Arduino Project Directory

> This Arduino project contains a program for testing the throttle and choke servos and calibrating their effective range. When run, it will repeating move each from "MIN" to "MAX" and back to "MIN". Change the constants at the top of the file to try different ranges and see which is right for your setup.

Assembly Instructions

1. Assemble the Adafruit Servo Shield
    a. Solder the included shield stacking headers to the bottom of the board.
    b. Solder the battery input housing to the top of the board.
    c. Solder the male pins to first four servo motor plugs on top of the board.
    d. Solder a male pin to Digital I/O Pin 3 on top of the board. (Outlined in purple in the wiring diagram.)
    e. Solder two male pins to V++ and GND Pins on top of the board. (Outlined in purple in the wiring diagram.)
2. Put Adafruit Servo Shield on Arduino. (Should stack on easily using headers; no solder or tooling required.)
3. Wire BEC
    a. Wire the BEC input to the Battery output
    b. Attach Y-cable the BEC output
    c. Wire one of the Y-cable outputs to the battery input on the Adafruit Servo Shield
4. Setup throttle and choke servos
    a. Plug the throttle and choke servos into motor shield ports 1 and 2.
    b. Download and run the latest Arduino IDE on your computer.

[1] Gitlab Repository URL: https://gitlab.oit.duke.edu/zjp3/generator-governer-control

   c. Install the Adafruit PWM Servo Driver Library from the zip file in the repository by clicking “Sketch > Include Library > Add .ZIP Library”.
   d. Upload the servo_test_utility program on the Arduino to test that the servos are working.
      The servos should move back and forth repeatedly. If they do not check your connections.
      STOP! *At this point the servos should be mounted on the frame with push rods attached to the engine control surfaces.*
   e. Adjust the constants at the top of the file servo_test_utility to restrict the servo range to that of the engine control surfaces.
5. Setup starter motor and ESC
      It is suggested that you complete the next two steps with the starter motor not mounted to anything. The two servos may also be disconnected.
   a. Solder male tips to the three output wires of the ESC
   b. Solder a male tip to the positive power wire of the ESC
   c. Solder a female tip to the ground power wire of the ESC
   d. Solder female tips to the three input wires of the starter motor
   e. Connect the three output terminals of the ESC to the three input terminals of the starter motor
   f. Connect the ESC control input (shorter of the two 3-wire plugs) to the Adafruit Servo Shield
      1. Orange wire connects to Digital Pin 3
      2. Red wire connects to V++ Pin
      3. Brown wire connects to GND Pin
   g. Connect the ESC to the battery.
      If everything is hooked up correctly, the ESC will now emit a descending tri-tone followed by 6 beeps. If it does not, troubleshoot using the ESC manual (link in the Appendix). If an ascending tritone is also heard after the six beeps, skip to step i.
   h. Program ESC throttle range using the directions in the manual. To mimic the input of an RC transmitter run the rc_mimic_utility on the Arduino.
      If the throttle range was programed correctly the ESC will not emit the original startup noises (descending tri-tone and 6 beeps) followed by an ascending tri-tone.
6. Test starter motor and ESC
   a. Unplug the ESC from the battery.
   b. With the Arduino connected to your laptop, open the Arduino Serial Monitor.
   c. Upload the generator_controller program onto the Arduino.
   d. Plug in the ESC.
   e. After the ESC emits its initialization tones, type “1” into the Serial Monitor input and press Send.
      You should see “System Ready” output to the Serial Monitor.
   f. Type “1” into the Serial Monitor input again to start the motor and set it to operational speed. Type “6” to stop the motor once operation has been verified.

If the motor does not run, troubleshoot using the ESC manual (link in the Appendix).

7. Initialize the system

   If you disconnected the servos in step 5, reconnect them now.
   <u>STOP!</u> *At this point the engine be should be assembled, mounted and tested.*
   <u>STOP!</u> *The starter motor should now be mounted to the frame, with its shaft attached to the shaft of the engine.*

   a. Disconnect the ESC from the battery.
   b. Attach the spark plugs from the ignition to the engine.
   c. Connect the free output of the BEC's Y-cable to the ignition.
      <u>WARNING!</u> *The engine is now armed! In the event of an emergency, disconnect the ignition from the BEC to stop operation.*
   d. With the Serial Monitor open, reset the Arduino. It should prompt you as it did before.
   e. Connect the ESC to the battery. Send "1" after ESC has initialized.
   f. Utilize commands (1-6) to move through the operational states (See Operating the Engine).

Operating the System

The system is operated via the Arduino Serial Monitor. Once it has been initialized (See Step 7, Assembly Instructions), send one of the following 6 commands to move between the operational states.

"1" : Cold Start
- Choke closed, throttle 40%, starter motor 40%
- Once the engine "pops" a few times move to Hot Start.

"2" : Hot Start
- Choke 50%, throttle 40%, starter motor 40%
- Once the engine starts move to Transition I.

"3" : Transition I
- Choke 50%, throttle 65%, starter motor 40%
- Engine is given gas. Once it seems stable move to Transition II.

"4" : Transition II
- Choke 50%, throttle 65%, starter motor 40%
- Starter motor is slowed. If engine seems stable move to Run.

"5" : Run
- Choke 50%, throttle 65%, starter motor 0%

"6" : Stop
- Choke open, throttle 0%, starter motor 0%

Results and Open Issues

Looking back, there are a number of design issues that presented themselves that could be avoided during the next iteration.

ESC Component

The ESC we purchased was designed for hobby applications. It came with a built in feature called called blocked rotation protection. This feature monitors the motor current to detect a jammed motor. When it senses resistance, it reacts by stopping motor rotation. In our application, the torque required to turn the gas engine shaft was being interpreted by the ESC as a blockage and so the motor would stop and go. Additionally, the ESC had a start sequence feature that caused it to slowly ramp up the motor speed from a stop rather than immediately jump to the input speed. Another issue was that the ESC required substantial initial setup, all of which was described through the inputs of an RC transmitter.

In order to work around this for the prototype, we were able to change the settings of the ESC to do a “hard start” instead of the default. Under this setting, the motor would slow down instead of fully stopping when faced with resistance and the ramp up time was greatly reduced. That being said, the performance was still less than ideal given the power of the motor and we believe it is because this ESC is simply not designed for this application. We recommend finding a general purposes ESC with little added features for the next design iteration. Choose an ESC that simply converts PWM input into motor current output and does not require initial range programming.

Frame Material and Assembly Precision

Another design issue that presented itself late in the project was the way the engine and motor were mounted to each other and to the frame. Both components must be mounted to the frame because each will be applying force to the other at different stages of operation. The precision of this mounting becomes very important at high speeds. If the two shafts are even slightly misaligned (in either position or angle), the high amounts of applied torque will cause a lot of damage and instability. During our second demonstration of the system, screws holding the shafts together began to come out because their angles were slightly off.

To remedy this we recommend two things. First, the engine and electric starter motor shafts should be attached to one another before either is mounted to the frame. Setting the angles of the shafts is extremely difficult, if not impossible, once the components themselves are attached to the frames. Second, with the two attached to each other, make careful measurements of how the 4 mounting holes on each side (base of engine and base of motor) are separated from each other in 3D space. The piece of the frame that they will be mounted to should be made of a rigid material that will hold the system still even with the motors at full speed.

Future Work

Regretfully, there are several items we did not have a chance to get to this semester. Here we have laid out some important tasks that need to be completed for the next design iteration.

1. Integrating a kill switch with the ignition that allows its power to be control via the Arduino. We have the part in the foundry but could not get it to interface with the Arduino. Some of the code for this is actually already written in the generator_controller program.
2. Integrating battery voltmeter and tachometer for autonomous operation. This would switching between operational states as laid out by the operational flow diagram (See Overview).

Project Timeline

This section provides our weekly project logs detailing how the project progressed throughout the semester.

Adafruit Motor Shield

Week of October 5th, 2015

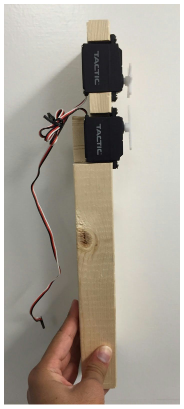
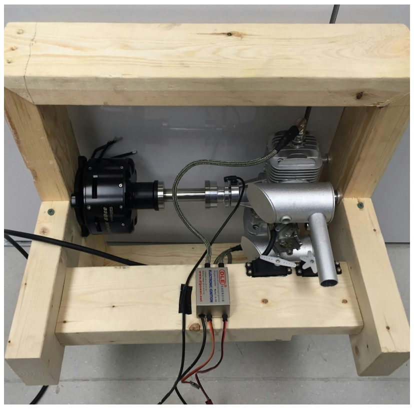
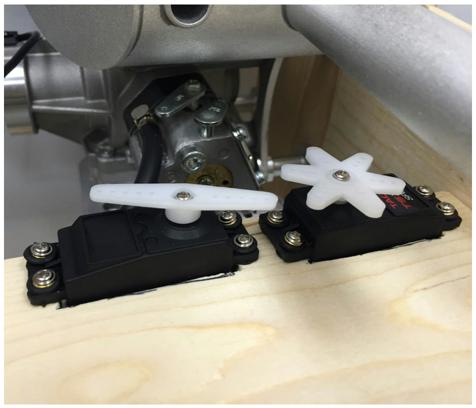
Before mounting the control servos to the engine, we wanted to test and configure the arduino setup. We configured an Adafruit 16-Channel 12-bit PWM/Servo Shield to operate the two control servos (throttle and choke) that would be mounted to the engine. See Video 1, Appendix A for demonstration.

Mounting the Servos

Week of October 12th, 2015

We built a new outward section of the frame to mount the throttle and choke control servos on. This section consists of two 8” beams on top and bottom and one 14” beam across. The cross beam has been notched so that the servos can be mounted inside (Figures 1 and 2).

Comments: The two beams attached to the existing frame ended up being too long. Would recommend 6.25” for future frames.

Engine Choke and Throttle Servo Integration

Week of October 19th, 2015

We connected the servos to the engine’s clutch and throttle latches by cutting and bending a pair of servo-pushrods to size. Next, we connected all of the engine governor components (battery, battery eliminator circuit (BEC), Arduino with Adafruit shield, and servos) to test the servo-pushrods. Finally, we calibrated the Arduino code to restrain the servos within the max and min positions for the choke and throttle. See Video 2, Appendix A for demonstration.

Starter Motor Integration, Sequencing

Weeks of October 26th, November 3rd

We integrated the code provided by the Generator Group to to coordinate throttle servo, choke servo, and starter motor ESC controls on a single Arduino. We wrote a new Arduino Sketch that assists in ESC and system initialization, and then allows the user to run 3 macro sequences: start, idle, and stop. Running a sequence will set proper values for the throttle, choke, and starter motor control components. Currently, the system requires an operator to interact with the Arduino via the Serial Monitor (at 9600 baud). See Video 3, Appendix A for demonstration.

First Demo, Troubleshooting

Weeks of November 10th, November 17th

The first demo was unsuccessful as the ESC did not seem to have enough torque to start the motor. After much troubleshooting we discovered that the ESCs bought come with a built in feature called blocked rotation protection. This feature tries to sense a jammed motor and react by stopping motor rotation. The torque from the gas engine was being interpreted by the ESC as a blockage and so the motor would stop and go. (The ESC was built for hobby planes, which have very little resistance.) By editing the settings of the ESC to do a “hard start” instead of the default we got it to a point where it will consistently start the motor within 20 seconds. See Video 4, Appendix A for demonstration.

Second Demo

Weeks of November 24th, December 1st

The second demo was a partial success. The engine started without any human interaction. However, it stop once the start motor power was turned off. Our theory is that turning the starter motor off instantly applies a large torque against the engine, causing it to stop. We are going to adjust the code to slowly ramp down the starter motor power while the throttle is slowly ramped up. We simply need to tweak the various levels throughout the operation lifecycle. See Video 5, Appendix A for demonstration.

Appendix

A. Video Links

1. Adafuit Motor Shield Demo: https://www.youtube.com/watch?v=uRKF8oibEU4
2. Engine Governor Servo Control Demo: https://www.youtube.com/watch?v=p3AZfDy_v4I
3. Coordinated Aurdino Control Demo: https://www.youtube.com/watch?v=PsbfKU4djPs
4. Improved Starter Motor Demo: https://goo.gl/photos/m8bjP6KYttizLTZN7
5. Successful Generator Start Demo: https://youtu.be/lLoOSBNVv2Y

B. Additional Resources

ESC Manual

http://www.hobbyking.com/hobbyking/store/uploads/443193686X1751013X31.pdf
User manual for the RotorStar 120A-HV ESC.

Getting ESC to work with Arduino

http://www.instructables.com/id/ESC-Programming-on-Arduino-Hobbyking-ESC/
Useful guide for getting the ESC to interface with Arduino. We've provided it here in case our advice is headed and a new ESC is used for the next design iteration.

Programming the ESC: (YEP is same manufacturer as ours)

http://www.hobbyking.com/hobbyking/store/uploads/893329926X10239X17.pdf
Used in conjunction with the provided RC mimic utility to setup the ESC "throttle range" and change/disable the operation settings.