Fall 2013: Electric Bike Boost Controller using the ATTiny85

Introduction

This paper covers the theory and practical use of a controller powered by an Atmel micro controller to operate a boost converter power driver used in an electric bike configuration.

A typical electric bike setup has a single voltage source to drive the motor from a controller. This presents a limitation because the battery size will dictate how fast and far the electric bike can travel. With a fixed voltage, the battery size must be specified. Smaller batteries are lighter, charge quickly, cost less, and are easier to service. However, they are limited in speed and distance. Larger batteries can operate and greater speeds and travel further distances, but are heavy, expensive, and take longer to charge.

Use of a boost converter is one option to get the benefit of small and large battery setups. The boost converter takes the voltage from the small battery and “boosts” it to a higher value, which is used to drive the motor at higher speeds. The converter is not 100 percent efficient, so some energy is used, somewhere in the range of 5-15% loss. This loss generates heat, so the boost converters are fitted with a heat sink to dissipate it from the device.

This project fills in the gaps by adding an output level controller that is operated from a switch mounted on the handlebars of the bicycle. When the operator presses the switch, the voltage boost is active. Another press of the switch returns the bicycle to the default voltage level of the battery. This is all done by means of an Atmel ATTiny85 microcontroller.

Project History

The mid-term subject originally submitted was a headlamp and turn signal control system. This project would not meet the deadline due to code complexity, hardware manufacturing, and availability of parts. The subject submittal process creates a commitment by the participants to responsibly take on the task and finish it in a timely manner. Since numerous issues surfaced late in the projects phase, a request was made to change the subject submittal.

By default, the change is not an option and any request to change is not approved. As a result, certain conditions were offered that would convey the sincere and responsible nature of the request. First, the choice of a term subject would now be the headlamp system. Normally the subject for the term is still open for choice until the first weeks of November. Next, the request would be mentioned in the mid-term paper, with details about the reasons why the project would not meet the deadline. Finally, the new project submittal would be presented within the defined deadline as the original submittal.

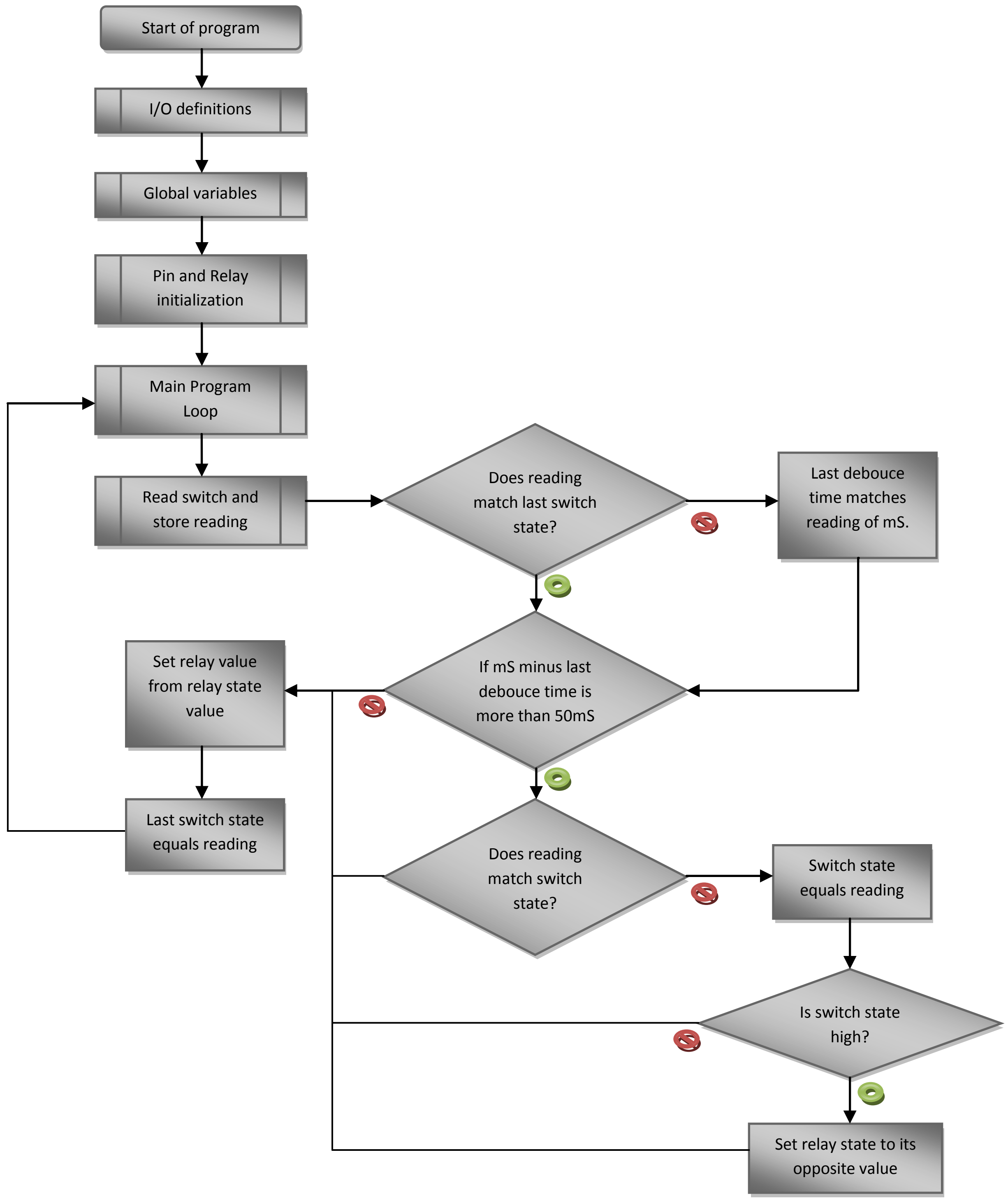
The change request was approved and as a result this paper is now on the boost controller system. In short, these were the lessons learned from the first submittal.

- Commit to the project, failure to will result in project failure.
- Create a working flowchart of the process early on.
- Create the schematic of components.
- Prototype the system and note missing components so they can be procured early.
- Simplify the code by use of code building blocks that can be integrated as a whole later.
- Focus on the results and identify how it accomplish them.

Program Code

```
/* -----[ Program Description ]---------------------
 BoostConverterController.ino
 Created October 14, 2013
 by Patrick Gilfeather

This program is for use as an electric bicycle boost controller.
The controller is operated by a switch mounted on the handle bars.
The operator pressed the button to enable the boost converter.
Each press of the button will toggle the boost converter on or off.
The ATTiny85 chip reads the switch state and enables or disables a
pair of relays on a daughterboard.
The relay units are change the circuitry of the boost converter
between normal (24 volt) operation to boost (42 volt) operation.
The operator gets the benefit of a small battery without the costs
of a large battery.


 */

// -----[ I/O Definitions ]------------------------
const int SwitchPin = 2;                // Pin 2 of ATTiny85 is connected to switch input
const int RelayPin = 1;                 // Pin 1 of ATTiny85 is connected to relay daughterboard

// -----[ Variables ]------------------------------
int RelayState = LOW;                   // Initial state of relay unit is set to low by default
int SwitchState;                        // Switch state variable
int LastSwitchState = HIGH;             // Last switch state is set to HIGH to allow default off
condition
long LastDebounceTime = 0;              // Initial last debounce time count
long DebounceDelay = 50;                // Debounce delay is set to 50ms so switch doesn't
accidently toggle from switch noise

// -----[ Initialization Code ]---------------------------
void setup() {
  pinMode(SwitchPin, INPUT);            // Sets Pin 2 of ATTiny85 to input
  pinMode(RelayPin, OUTPUT);            // Sets Pin 1 of ATTiny85 to output
  digitalWrite(RelayPin, RelayState);   // Writes the pin state of Pin 1 from initial LOW state
}

// -----[ Main Routine Code ]---------------------------
void loop() {                                              // The function loops endlessly...
  int reading = digitalRead(SwitchPin);                    // Takes a reading from Pin 2 and sets
the value of the new variable
  if (reading != LastSwitchState) {                        // If the reading doesn't match the
last switch state value then do the following:
    LastDebounceTime = millis();                           // Set the value of the last debounce
time from the milliseconds read by the ATTiny85
  }                                                        // Otherwise, just continue on...
  if ((millis() - LastDebounceTime) > DebounceDelay) {     // If the millisecond reading is more
than the debounce delay of 50ms, then do the following:
    if (reading != SwitchState) {                          // If the reading from Pin 2 doesn't
match the variable, not set by default, then:
      SwitchState = reading;                               // Set the switch state variable to the
value of the reading
      if (SwitchState == HIGH) {                           // If the switch state is high, then:
        RelayState = !RelayState;                          // Then the relay state should be the
oposite value of itself
      }
    }
  }
  digitalWrite(RelayPin, RelayState);                      // Now Pin 1 is given the value of the
relay state
  LastSwitchState = reading;                               // Lastly, the last switch state value
is set to the value of the reading variable
}
```

Operational Flow Chart


Start of program
I/O definitions
Global variables
Pin and Relay initialization
Main Program Loop
Read switch and store reading
Does reading match last switch state?
Last debouce time matches reading of mS.
If mS minus last debouce time is more than 50mS
Set relay value from relay state value
Last switch state equals reading
Does reading match switch state?
Switch state equals reading
Is switch state high?
Set relay state to its opposite value

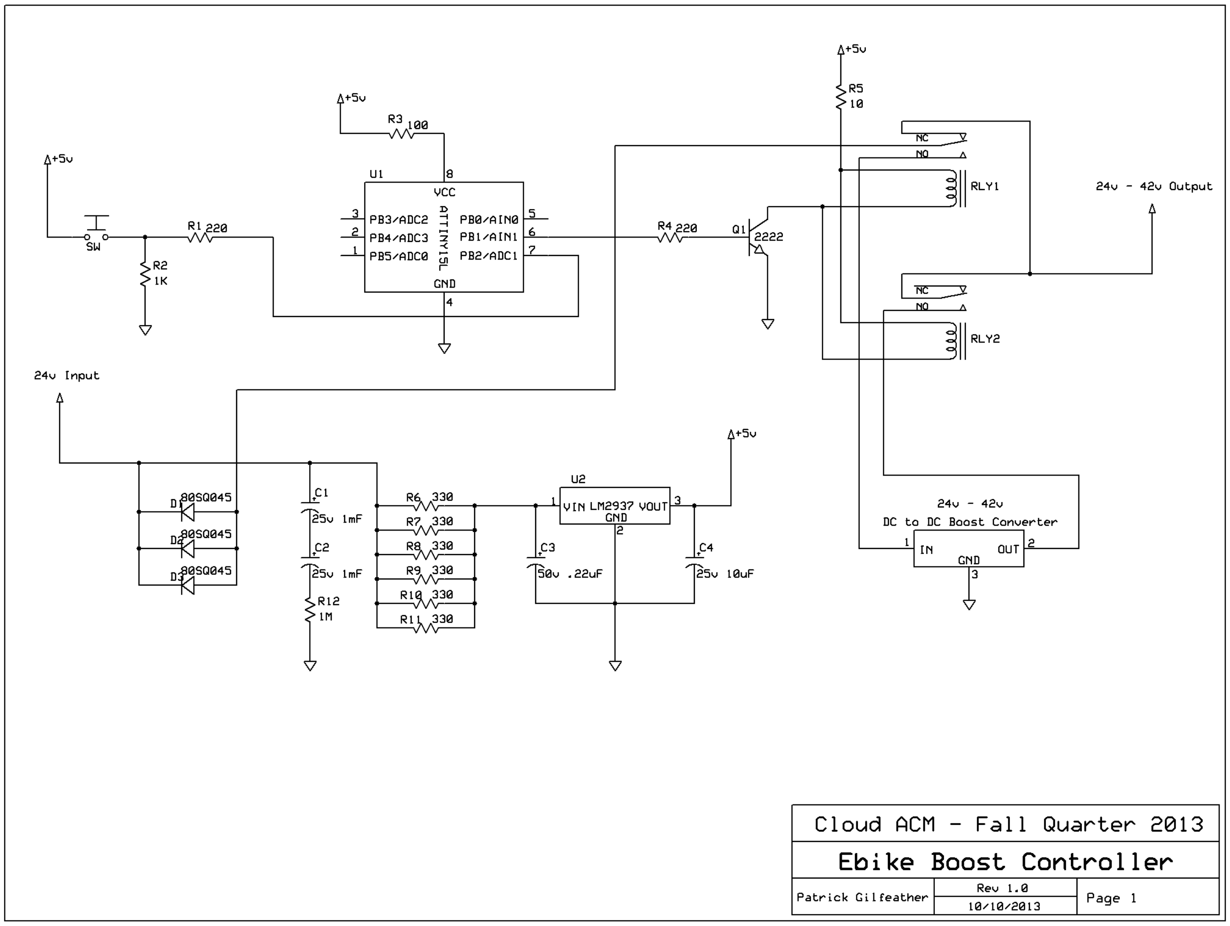
Theory of Operation

The operation of the control circuitry had to be powered from the electric bicycle battery source. This requirement simplified the functional operation of the controller by the operator, but added circuit complexity as a consequence. The power requirements of the ATTiny85 microcontroller and the relay daughterboard matched. As a result, only a single regulated voltage of 5vdc was required to operate the control circuitry. A spark prevention buffer was placed on the supply input to prevent damage to the relay connectors. Also, a bank of resistors was placed on the input of the 5vdc regulator to prevent over heating from the relay driver loads. Lastly, a bank of schotky diodes were placed in series to the 24vdc supply to prevent reverse voltage feeds from the boost output, should the relays not operate in sync.

When the boost controller is connected to the source voltage, the device will energize. The ATTiny85 will begin to execute its boot loader and run the stored program code. By default, the relays are not on and the output of the boost converter should match the input source of 24vdc. Operation of the electric bicycle motor should reflect the source voltage. Drain on the source voltage from the boost converter should be extremely low if detectable at all.

When the boost switch is pressed from the handle bar cluster, the ATTiny85 will read that value and trigger on the relay daughterboard. This will change the 24vdc circuitry and route the 24vdc into the boost converter module. The output of the boost converter module will also be routed to the output of the boost converter unit. The relays will hold the on state after the switch is released on the handle bar cluster and will continue to until the button is pressed again. The drain on the source voltage while the relays are active and the boost converter module is energized is roughly 5-7watts.

Schematics of EBike Boost Controller and Devices



Bench Testing Results

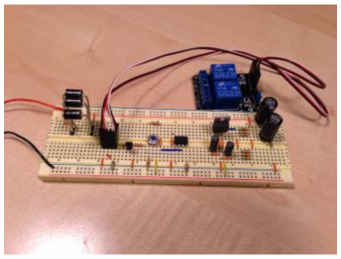
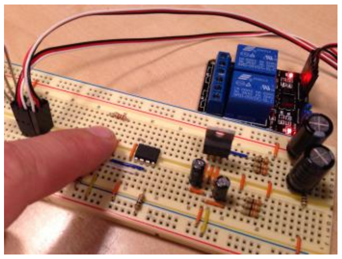
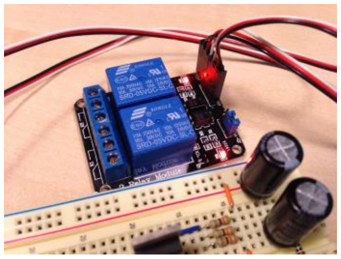
The bench test was limited since most of the components were missing. The hub motor and boost converter unit were not included, since the relay states could be verified without these components.

One of the key aspects of the bench test was a bread board prototype. Having the bread board, in addition to the finished prototype, helped in development and testing. It was concluded that any development in the future should require a working bench platform for reference, testing, and validation.

The logic circuit, which comprised of the ATTiny85 and 2 relay module, was fairly straightforward. Since the relay module required an active ground to enable the relays, a NPN transistor driver was used to perform the switching. The handle bar switch is a pull up type, this uses lower power when not in use.

The rest of the circuit was the power systems. A resistor bank helped limit current through the 5vdc regulator that powers the relay module and ATTiny85 microcontroller. There is a spark prevention circuit to limit sparks while connecting the controller to source power as well as to protect the relay contacts from surges. Lastly, there is a schotky diode bank to prevent boost voltages back to the source voltage. The back voltage difference could damage the voltage regulator.

The prototype operated as expected. Heat was a concern for the 5vdc regulator. It would be recommended that the final prototype have a heat sink on the regulator. There was a noticeable voltage drop on the schotky diode bank. It was later discovered during field testing that the diode bank had to be bypassed to allow operation of the boost converter controller.

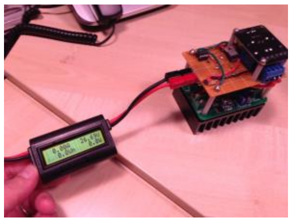
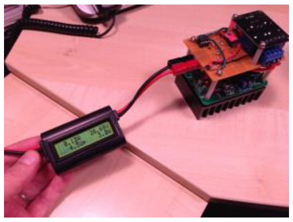
Field Testing Results

The final prototype was mounted on the boost converter module, with the relay module mounted above. The standoffs used were of the same threading as the boost converter module heat sink, which was M3 course threading. Nylon spacers were also used to insulate the standoffs and provide a washer function to stabilize the circuit boards.

The enclosure is a LMB Heeger Tite-Fit chassis. The dimensions are width 4” x length 5”x height 3”. The final prototype was mounted to the chassis via the heat sink of the boost converter module using nylon spacers. Holes were cut in the chassis to allow the Anderson connectors and control switch access. Slots were also cut in the side facing the heat sink to allow the device to remain in a safe operating range.

Testing of the final prototype revealed some issues. First, the schotky diode bank was preventing the unit to operate the relay module. The diode bank had to be bypassed in order for the boost converter to operate. Second, the watt meter display would go blank, or show corrupt characters when the boost converter was cycled through its modes.

The test results concluded as expected. The motor would operate at a normal range when the boost converter was not activated. When the button on the handle bars was pressed, the motor immediately increased speed and the watt meter reflected the voltage increase. The control button operated as expected, the modes would switch and remain until the button was toggled again.

The final prototype operated within a normal temperature. The current and wattage of the boost converter during activation was identical to the readings taken from the bread board prototype.

Summary and Conclusion

This project demonstrated the use of an ATTiny85 microcontroller to control modules from operator input. The specific goal of the project was to control the voltage fed by a limited source and increase the speed of an electric bicycle. These objectives were accomplished.

The project subject submitted was changed due to issues that were not foreseen until late in the project phase. There were key points to why this occurred:

- Failure to commit to the project.
- No work flow of the process existed.
- No schematic of the design was available until later in the project.
- The prototype was created late in the project phase
- Too much emphasis was placed on the code and how it would work.
- Failure to focus on the results and identifying how it accomplish them.

Once the project subject change was accepted, the new project had just over 1 week to deliver. Fortunately, the code involved in the project had been covered and a bulk of the circuit had been developed.

The final prototype was a particular challenge because it was a first of its kind. The robust construction is a direct result of previous failed attempts to use bread board prototypes in the field.   The modular nature of the components made the final prototype small and manageable to install.

This was a challenging project and the successful operation of the final prototype is rewarding. The length of operation and reliability can only be determined over time. This is only the start of the this project.