Pedal-Powered Smart Bike

ECE 445 Points Tables

Alex Sirakides, Karl Kamar, and Anshul Desai
Team 29
TA: Yifan Chen
November 13th, 2020

Our listed high level requirements:

1. The user produces at least 9V by pedaling in order to charge the 8V battery. That is 100RPM for the bike tire. **(covered in the power module points)**
2. LCD interfaces with the GPS module and displays the user's speed, distance traveled, longitude, latitude, current time, current date, time of ride at a refresh rate of at least 1 Hz (1 update per second) upon time/location fix.
3. The ON/OFF button must open the circuit at the battery output, and Reset button must reset distance traveled and ride time to 0. **(control module)**
4. Turn-light buttons must make turning lights blink at a rate of 60 flashes per minute at least when pressed and turn them off when pressed again. **(control module)**

Power Module

Requirement	Requirement	Points
Pedals need to turn with no more than 5N-m of torque as user input.	Due to the nature of pedaling a bike (push pull interaction) measuring a constant force will be difficult. Since the end product here is that the torque requirement means it can be pedaled, this can be verified by inspection.	
Generator must generate 9 volts of power.	Prop up the back tire so that the bike may be tested in place with a voltmeter wired in parallel between it and the charge controller. 9V is the target voltage.	
The generator rod must rotate with the tire.	While the bike has its back wheel propped up for the motor verification, the wheel will be watched to see if consistent contact is made. If consistent contact is made, >90% of the time, the rod is effective.	
Has a diode preventing current flowing from battery to generator.	If current flows in the opposite direction, our generator will become a motor. Therefore, we can verify this by seeing if the wheels turn without user input when the battery is connected.	
Prevents the battery from overcharging	Charge Controller 7-segment displays voltage at the battery terminals. Charge should stop flowing to the battery after it reaches its max voltage (9.6V). Verifiable by hearing the tick from the charge controller which is the relay disconnecting the battery from the motor. The display instantly shows a reduced voltage value because the battery is not charging anymore. Same process when charge is starting with min voltage (7.2V)	
Needs to have the capacity to power up the circuit.	System powers up when the on switch is flipped, while battery is charged. Max capacity of 3.2Ah. Duration and capacity values in datasheet	
Steps down 8V battery input to 5V output for the rest of the system.	Use multimeters to test input and output voltages of the buck converter.	
Total		**15**

GPS Sensor Module

Requirement	Verification	Points
Refresh rate ≥ 1 Hz	The default baud rate of 9600bps gives us an update rate of 1 Hz. With a max baud rate of 230400, this gives us a frequency past what is required. Using the Serial monitor of Arduino IDE, we tested data next to time stamps to see how many updates per second we get for our different baud rates. The default 9600bps gives us one update per second and we set our baud rate to 304800 to allow for more updates for more accurate speed measurements.	
MCU does not interface with garbage metrics from GPS Module	We ensure that garbage data doesn't feed into our calculations or LCD display. We set up multiple conditionals to ensure that any odd occurrences of data from the GPS sensor get ignored. For example, if our current latitude or longitude is equal to 0.0f, then it is likely that it is garbage data. Therefore, we do not consider this location when calculating total distance, average speed, or the displayed coordinates on the LCD.	
Total		**5**

Display Module

Requirement	Verification	Points
Microcontroller buffers display correctly	The requirements for the LCD display is that the SRAM from the microcontroller is at least 1KB. This is a minimum requirement in order to hold our display buffer. Our MCU has an SRAM of 2KB, but we can verify that the MCU can buffer the display by testing our template which updates every second.	
Display updates at rate of 1 Hz.	Since we are expecting updates of once per second from the GPS Module, we want our display to reflect these updates. We therefore require updates every second on the LCD display. This is ensured by only clearing and drawing the video buffer when our serial connection to our GPS module is available, and this only occurs at our refresh rate of 1 Hz defined above. We verified by printing to our Serial monitor with a timestamp in Arduino IDE every time the video buffer was drawn	
Display shows heading, time, speed, current speed, total distance, ride time, and coordinate location when GPS gets relevant time and/or location fix	When we only have a time fix, we can only increment our current time and ride time. Speed, distance, and location information happens after a location fix. This was tested when we first purchased the GPS sensor, as the LED indicator flashes 1 Hz when there is a time or	

	location fix. Again, we ensure to filter our garbage data using conditionals. GPS fix behavior was observed through inspection in testing the complete product.	
Total		10

Control Module

Requirement	Verification	Points
Reset switch must reset data when un-depressed and record data when depressed.	Test by inspection. Unit testing performed on Arduino IDE.	
ON/OFF button must power the whole user interface off by disconnecting the battery from the rest of the circuit	Test by inspection.	
Turn Light Buttons must make turn lights flash at a rate of 60 flashes per minute. Must turn them off when pressed again.	Test by inspection.	
Headlight and taillight turn on when the ambient light sensor detects low visibility.	Turn off lights and check whether the bike lights come on. Unit testing performed on Arduino IDE as well as with a button and LED to simulate ALS/FET.	
Headlight and taillight turn off after 30 seconds with the ambient light sensor detecting sufficient visibility.	Turn lights on and check whether the bike lights turn off. Unit testing performed on Arduino IDE as well as with a button and LED to simulate ALS/FET.	
EEPROM saves exercises statistics according to the following conditions: 1) Ride time only incremented when GPS has time fix 2) Average speed and distance only calculated when GPS has location fix 3) When device is off, this duration of time is NOT factored into any of these three	State machines to preserve statistics as well as any relevant values to the EEPROM. Upon test by inspection and unit testing on Arduino IDE, we were able to verify that everything is saved correctly.	

statistics.		
Average speed only calculated for speeds > 0.5 mph and total distance only calculated for speed > 1.0 mph	Conditionals included in each relevant function to check current speed and ensure that it is within bounds. Put in place due to wandering behavior of GPS module. Unit tested on Arduino IDE.	
Total		20