**Team Member Responsibilities:**

Christian Alcalde:

* Proximity sensor drivers
* Obstacle detection and avoidance programs
* Power

Abdullah Wardak:

* IR sensor drivers
* Path follower
* CPU

Eric Fong:

* Motor control hardware and drivers
* Accelerometer drivers
* Positional feedback mapping w/ accelerometer

**Project Name:** Autonomous Car

Our main goal is to create a car that follows a custom path made from black electrical tape. It would also have a method for avoiding obstacles and moving past them using a recursive function that allows the car to make several circular arcs around the object until it detects the path again. Power will be provided to the car through a 4 AA battery holder (equivalent to 6V) connected to the processor. Non-rechargeable batteries will be used in junction with a voltage regulator to supply 5V to the PCB VDD instead of the original 6V. A switch on the car will turn the car on and off, and a button will be connected to the CPU as a reset button. The motors we have listed on the bill of materials draw 750 mW each at load rating.

A simple project car chassis will be the base of our vehicle. It comes with two wheels and two DC motors, although we may use other motors if the included ones do not have good documentation or specs.

Path following will be implemented using three IR reflective sensor modules, the CPU, and the motor driving subsystem. The IR sensors will observe reflective values of three spots in front of the car, one in the middle and one each to either side, and send readings to the CPU. If the CPU detects a change in readings from the sensor, it will execute one of two turning functions until the sensor output returns to the on-path values, then resume forward travel. A dark signal from the left sensor will encode for executing a left turn function, vice versa for right turn, and a dark signal in the middle with bright on the sides will encode for the on-path state.

The motor driving subsystem will control the speed at which the motors run and the direction they move the vehicle in. Steering will be approached by running the left and right side motors at equal speeds in opposite directions to rotate the car in place. To implement this, we will have the CPU send control signals to a L293DD H-bridge IC which can change the polarity of current through the motors. These same connections can be used in junction with a PWM signal to vary the average voltage supplied to the motors and produce variable speed control for the motors.

Obstacle detection will be implemented using an ultrasonic distance sensor to detect any objects in the way of the car. If the CPU receives a signal from the sensor indicating an obstruction is within three inches of the sensor, the car will activate the obstacle circumnavigation subsystem.

Obstacle circumnavigation will be triggered by obstacle detection, and will be implemented using the travel logging, obstacle detection, and motor driving subsystems on the CPU.

For a base implementation of a circumnavigation function we may assume that all obstacles require no more than about a two-meter-long detour and can be circumvented in a semicircular arc or series of at most five such arcs. More complex pathing algorithms would be a stretch goal. Basic obstacle circumnavigation would be achieved by executing a recursive function that attempts to drive the car in a ~30 cm long semicircle of radius ~9 cm around the obstacle. If the car does not intersect with its original course or encounters another obstacle while circumventing, then the controller will recursively call the algorithm until it’s back on track, from which point the car will go back to following its plotted course. If the car can’t get back on its original course after a 2 meter detour, it will stop and flash a failure message on the 16x2 LCD character display. It is important to note that this algorithm is limited to objects that are 2ft by 2ft.

Travel logging recording and display would be facilitated using the CPU, accelerometers, oscillator, 16x2 LCD character display, and some buttons. The accelerometers will feed data to the CPU on the oscillator-supplied clock cycle to log the acceleration-time of the vehicle into a data structure in memory, and this data will be fed into an integration algorithm to obtain velocity data, which will also be fed into another integration algorithm to obtain position data. Professor Tandon has indicated that we should seek consultation on the choosing and usage of an oscillator.

Travel metrics including total distance traveled, average speed, and average acceleration will be calculated from the acceleration-velocity-position data and be outputted on the display. displayed metrics will be scalar quantities calculated from the magnitudes of the x-y acceleration, velocity, and position logs. The display control and decoding will be handled by the CPU and onboard controller of the display which communicate over a parallel connection. The user will be able to cycle between which stat is displayed by pressing a button linked to the CPU.

Base Objectives:

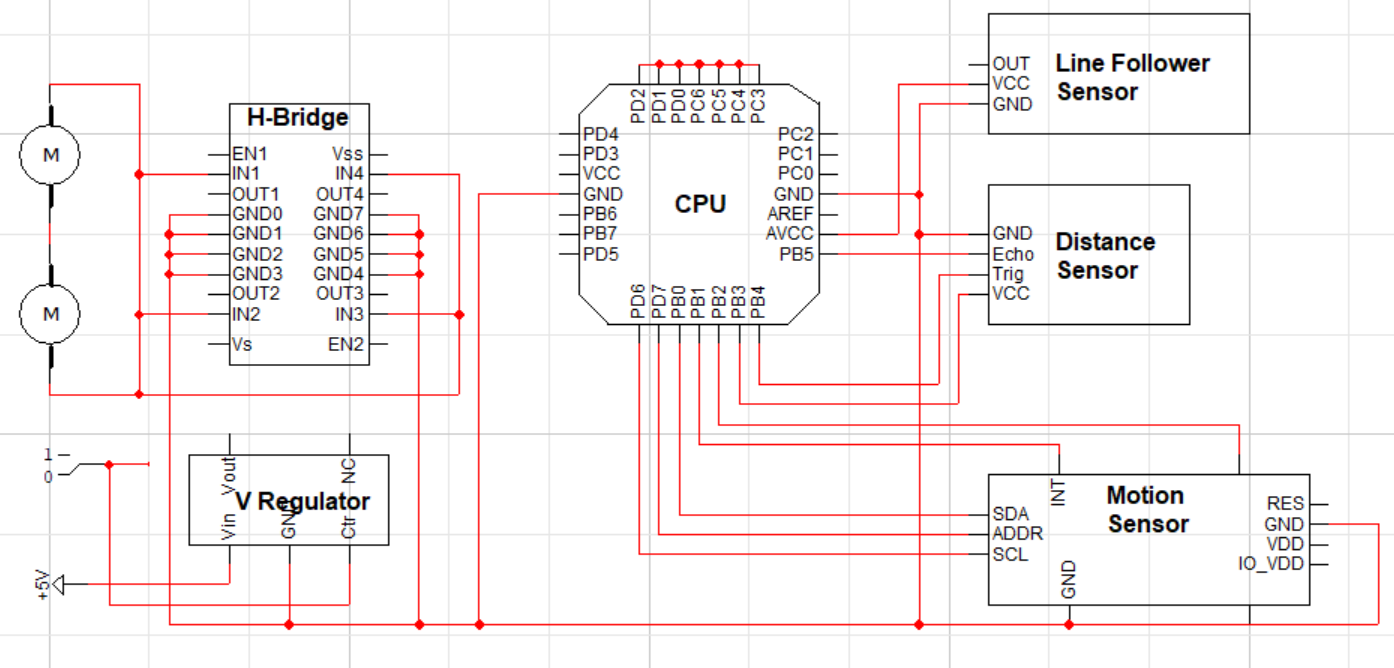
* car - controls motor steering, speed, and direction
* electric-tape path following
* obstacle detection
* obstacle circumnavigation
* travel statistic logging and display
  + each accelerometer can detect acceleration in three axes
  + accelerometer logs acceleration data
  + CPU integrates acceleration data to obtain velocity data
  + CPU integrates velocity data to obtain position data.
  + data for each axis is logged in its own data structure instance
  + displayed scalars calculated from magnitudes of x-y data
  + Viewable from 16x2 LED character display
    - Cycle through average acceleration, average velocity, distance traveled with the push of a button

Stretch Goals:

* advanced obstacle reaction
  + get accurate obstacle readings while car is in motion
  + detect if obstacle is moving or stationary
  + back up from approaching obstacle
  + wait out through traffic
  + add another motion sensor, detect moving obstacles via triangulation/sensor differences, avoidance swerving
* log exporting via serial UART cable
  + exporting functions triggered with a button press when plugged in
  + draw a map using x-y position data from a run and export the picture



For motor driver IC, which drives the motors we will be using (DC brushed) H-bridge.



* spare CPU pins to be allocated for debugging pin headers, leds, buttons, etc. as available