Self-Timing Competition Switch

Jakub Nejedly

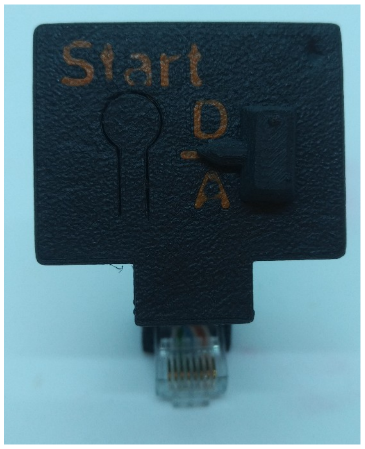
Andy Nilson

BARCBOTS – Speed Demons

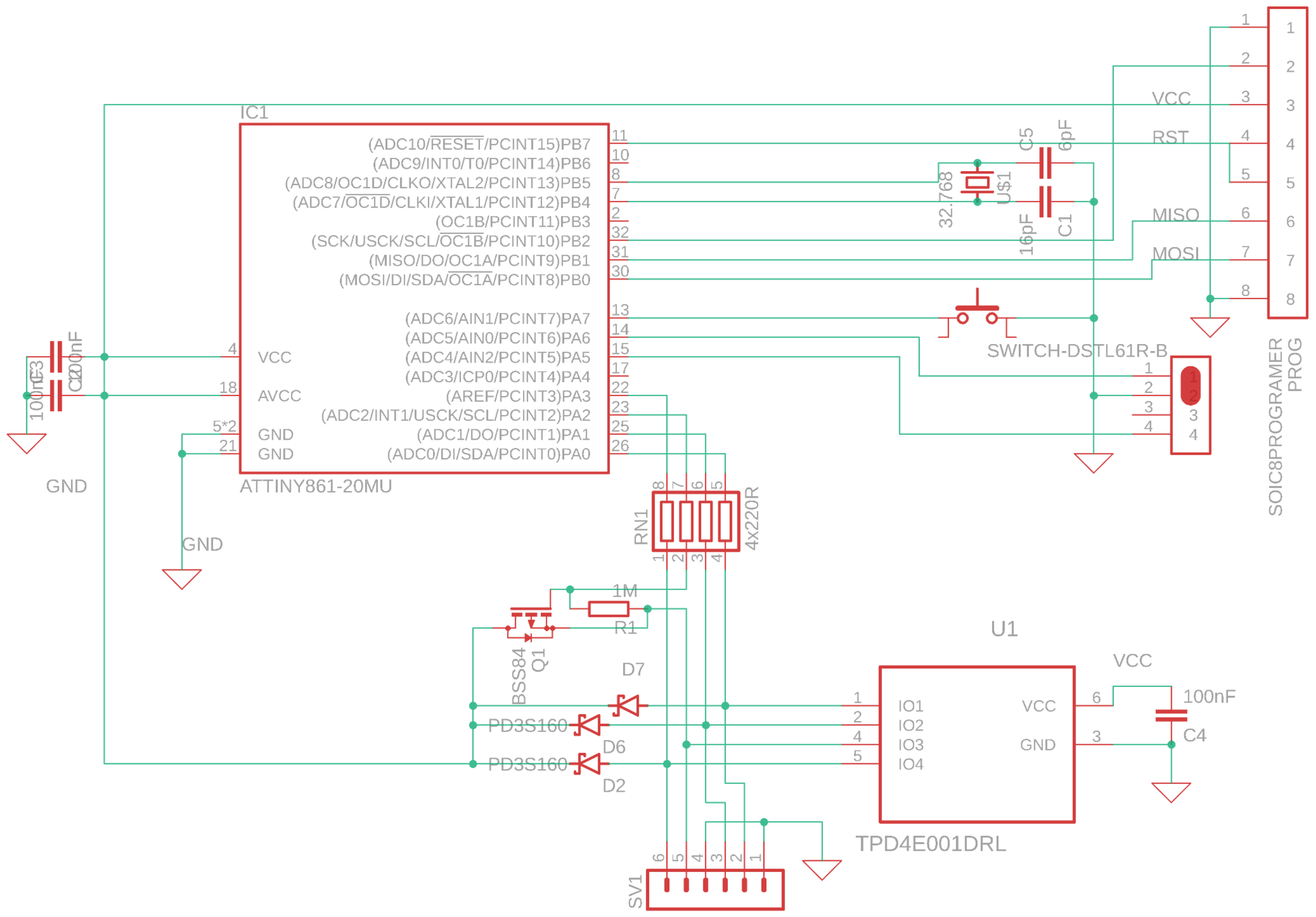
Team: 11101A

Cupertino, California, USA

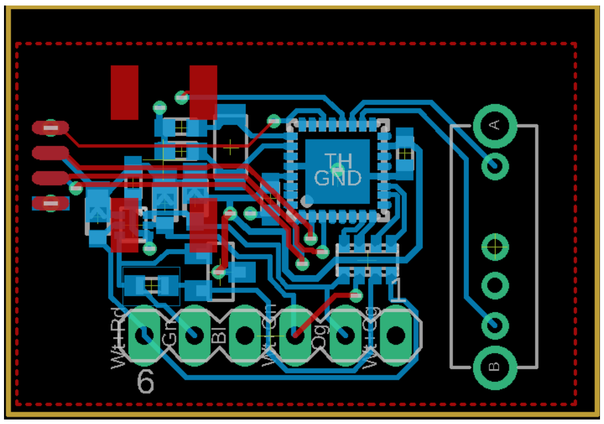
Our team designed a self-timing competition switch. It is a small box that clips onto a controller using the RJ-45 jack on the back, and a small hook that latches onto the edge of the screen. When connected, it enables the user to run a competition length timed driver or autonomous period. The user first selects which mode the switch is in. There are 3 modes. The first is marked with a D, and this is the driver mode. When the mode is selected, the robot is disabled, until the button is pressed. Pressing the button enables Driver control for one minute and forty-five seconds. Pressing and holding it for half a second enables the driver skills challenge, which runs for only a minute. The next mode completely disables the device, allowing the robot to function as if no competition switch was connected to it. The third and final mode is the autonomous mode, which functions the same way as the driver mode – a press starts a fifteen second autonomous period, while a hold starts a one minute autonomous skills period.



The creation of this competition switch was something we found necessary when we were programming our robot. While the controller is able to time itself when running a program, we had some issues with that. Every competition requires your robot to be able to do several different autonomous routines, so we coded in a selection process for our various autons at the start of our program. The problem with this was that the built in controller timer would bypass our selection process, so we would always run the default program, not the one we wanted to test. It also would start the driver period right after the autonomous, giving no time to evaluate how the auton did. Along with that, when we use the built in timer, the controller ends the program when the time runs out, so we cannot check any telemetry dumped to the screen during the run. We used to run the autons by using the manual competition switch and timing it, however our autons were often within 100s of milliseconds of the cut off time, making it hard to tell if it had been on time. Our competition switch is easier to use and more capable than the timer built into the controller.



The project, like any good project, started with a breadboard and volt-meter. We had to decode which signals in the RJ-45 jack did what. The first thing we built was just a manual competition switch, to make sure our map of the signals was correct. Once we had that knowledge, we were able to create a schematic for our competition switch. There were several things we needed to do. The first was to power the thing. Through previous testing, we knew that an ATTiny-861 clocked by an external crystal would only draw a small current. Along with that, we knew that there needed to be some power coming from the controller itself for the normal competition switch to be able to work. We created a small circuit that allowed us to siphon off the power “leaking” through the data-pins to power our processor. We also connected some GPIO pins back to the data pins on the controller, and some to the mode selection switch. This allows us to control which pins were active through the microchip. We created both the schematic and the final board design in AUTODESK Eagle. Since most of the parts we used were not in the existing libraries, we created our own library for them, with the schematic symbols, board footprints and also 3D packages, which were created in



Fusion 360. Since this project has been in development for a while,

the newest version of Fusion 360 was always used, as it was updated and improved. Thanks to this library, and the unique integration between Fusion and Eagle, we were able to export the board directly into Fusion, with a replica of all the parts modeled onto it. This allowed us to design a well-fitting case for the electronics. We were also able to design the clip of our competition switch by using boolean operations to cut out the shape from a model of the controller. There were some problems with this, since the boolean operation was being done on a hollow body with imperfect seals, which resulted in weird structures, such as zero width walls. To solve this, we used the sweep tool, with a guiding rail, to create extra bodies which filled in the gaps and fixed our issues. We also wanted some markings in a different color on the face of our switch. To do this, we designed them as a seperate component, which we then exported as a single mesh. Once we had that, we sliced the Gcode for both the main body and the markings, then modified the Body Gcode by adding a filament change call after the first layer. We then printed the 2 layers of color for the text, and changed back to the main body color. Finally, we wrote a short program in the Arduino IDE which would define when our microchip would activate and deactivate the control signals, based on the inputs from the button and slider switch.

Our team learned a lot about creating electronic circuits, which is outside the regular framework of Vex robotics, even though it is tightly connected to it. We also learned about some limitations of CAD software, and had to design creative workarounds for them. We gained a lot of experience in different workflows for 3D modeling, as the design process for all the various electronic parts was very different to the process of designing the body of the competition switch. Finally, we learned the importance of perseverance, as we iterated through multiple versions of the design.