# Offloaded Odometry: Interesting Notes and Tidbits.

By Kunal

## Hardware

**ATMega32u4 on PJRC’s Teensy Board**

PRO: Hardware USB Support

CON: No FPU

**Two Quadrature Encoders**

On our robots, I calculated the interrupt rate to be (very approximately) 40kHz when both encoders are set to a resolution of 1000 PPR and are read in 1X mode. My test setup is significantly slower.

## Software

In my Excel simulation, I showed that cartesian position tracking is more accurate at higher update rates. Thus, my main objective was to run vector summation as fast as possible. In 2018, we updated our position tracking algorithm at 0.2 kHz. With offloaded position tracking, we can now update at 6.7 kHz.

**SIN(x) Performance**

I was taken aback by how slow floating-point operations are on the microcontroller. Trigonometric functions are even worse. To improve performance, I wrote a script to generate the C source code for a SIN lookup table.

TODO: I implemented my lookup function poorly. I use recursion when . Thus, the update frequency depends (±1kHz) on the quadrant of robot bearing. I’ll fix this soon.

TODO: I should try using only fixed-point math.

TODO: The ATMega32U4 can transfer data at ≥ 115200 baud. I could try encoding each tick as two bits (side and direction). Then, I could take advantage of the RoboRIO’s more powerful processor.

|  |  |
| --- | --- |
| **Setup** | **Update Rate (Stationary, Quadrant 1)** |
| Standard SIN/COS functions | 3.2 kHz |
| Lookup Table (~6000 value max) | 6.7 kHz |
| RAM Lookup Table (~500 value max) | 6.9 kHz |

NOTE: Changing the floating-point datatype (e.g. , , ) has no impact on performance because all floating-point types are 4 bytes long.

**Communication**

To (re)program the microcontroller, I need to plug it directly into my laptop. This could be very inconvenient on the robot. Thus, values like data rate, sensor inversion, and track length must be configurable by the host device. I’ve made my own host/microcontroller communication protocols before, but this was far more complex. Variable packet size, endianness, and Java’s lack of support for unsigned data made communication very difficult.