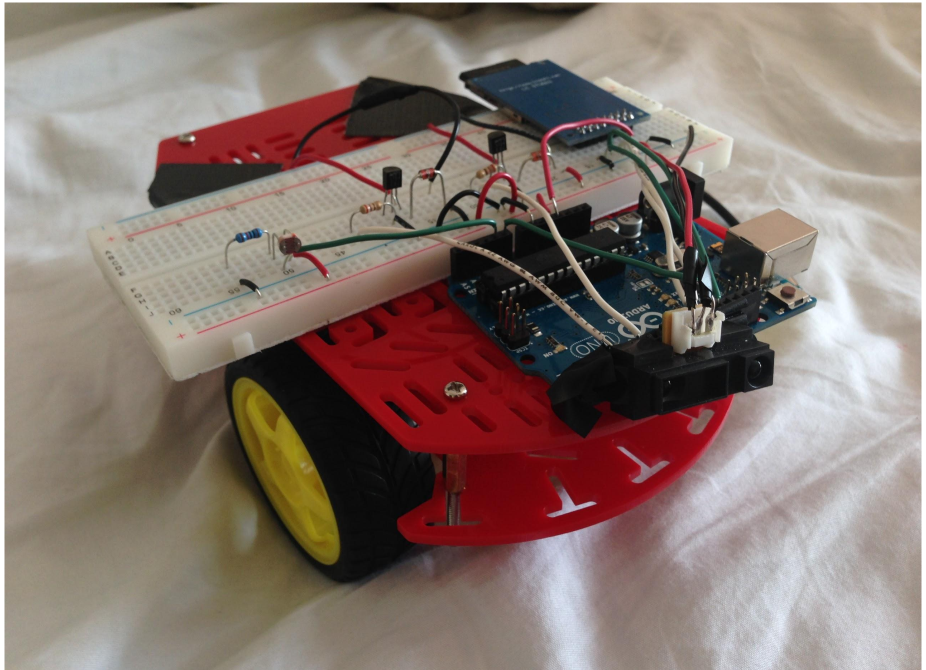
Embedded Systems

Rachel, Shane, Amanda, Mike

Our System

- Arduino-Controlled, autonomous vehicle
- Navigates room collecting data and saving it on an SD card



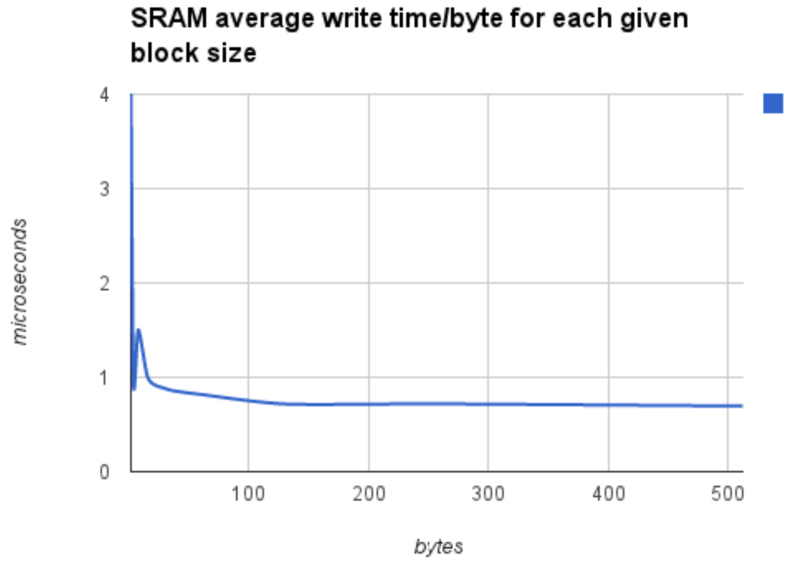
Arduino Uno (ATMega 328) onboard memory options

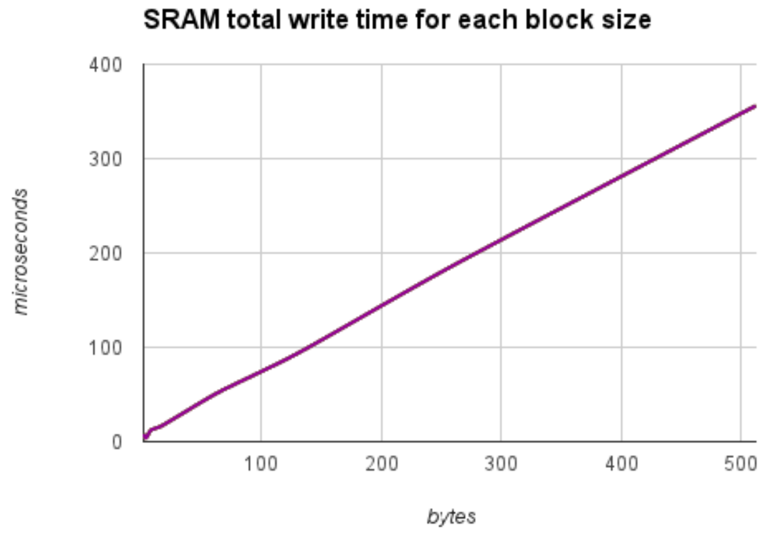
- Flash - nonvolatile 32k bytes where arduino sketch is stored
  (can be written to ~10,000x)

- SRAM - volatile 2K used for program variables
  (can be written to ~infinite)

- EEProm - nonvolatile 1k that can be used for program variables
  (can be written to ~100,000)

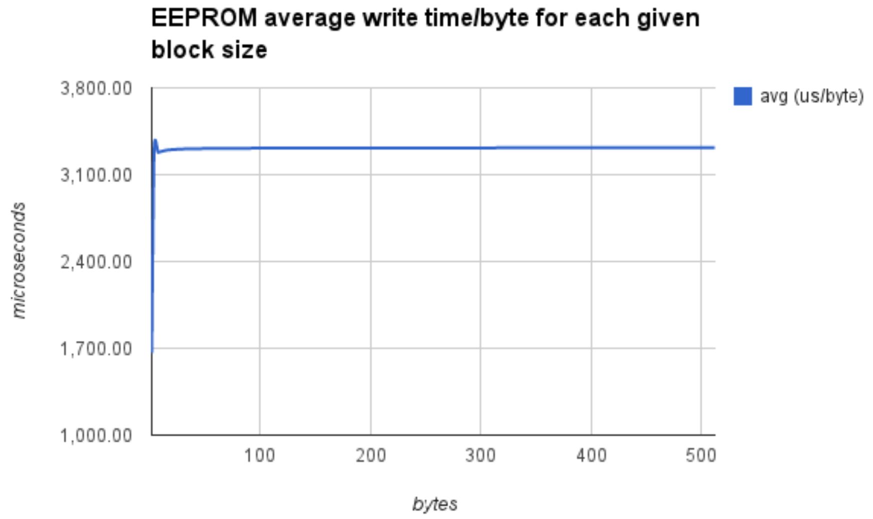
SRAM write speed tests

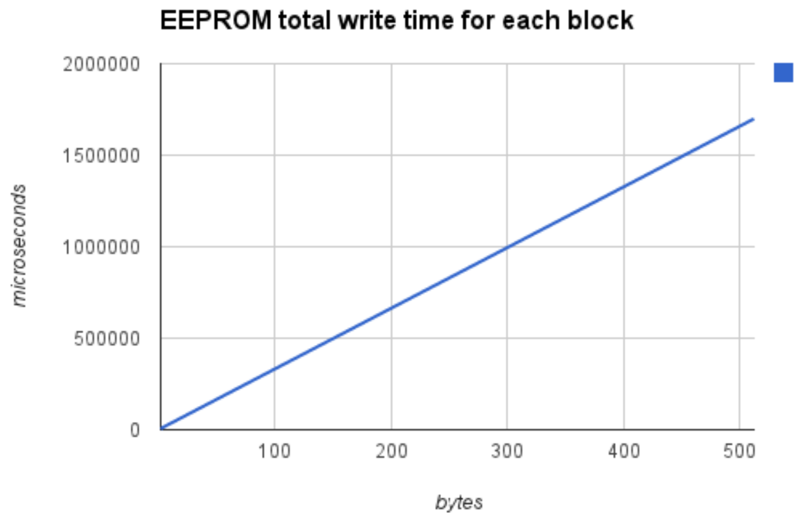




~1.25 us/byte

EEProm write speed tests

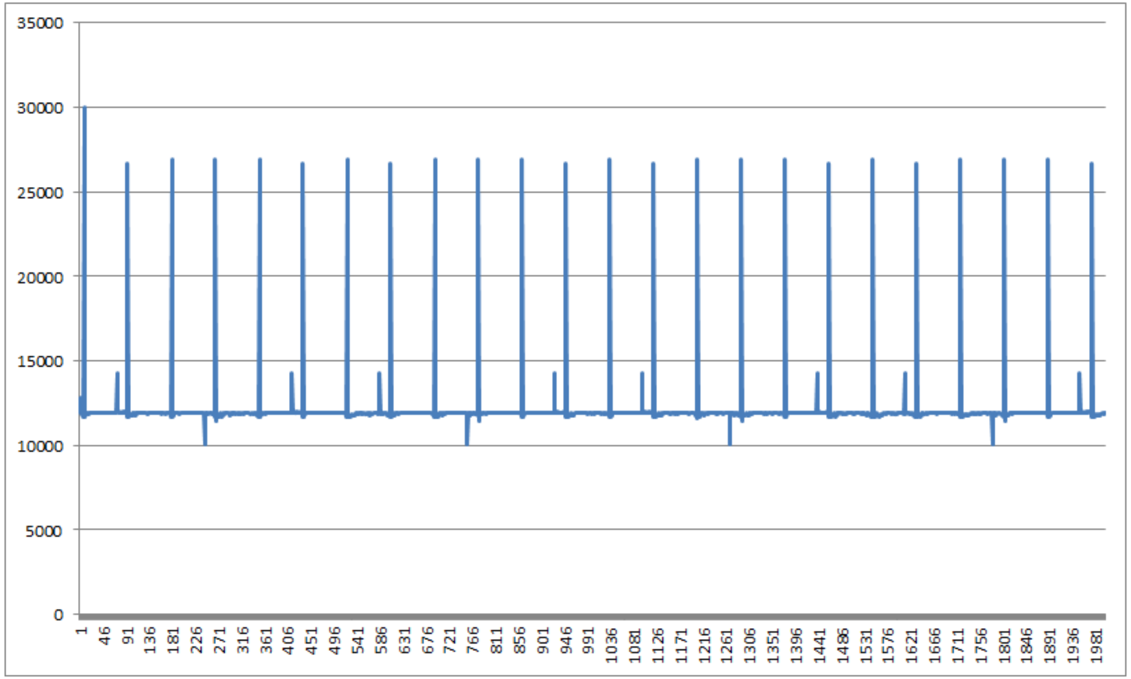




~3100 us/byte

Writing single bytes to SD card

microseconds

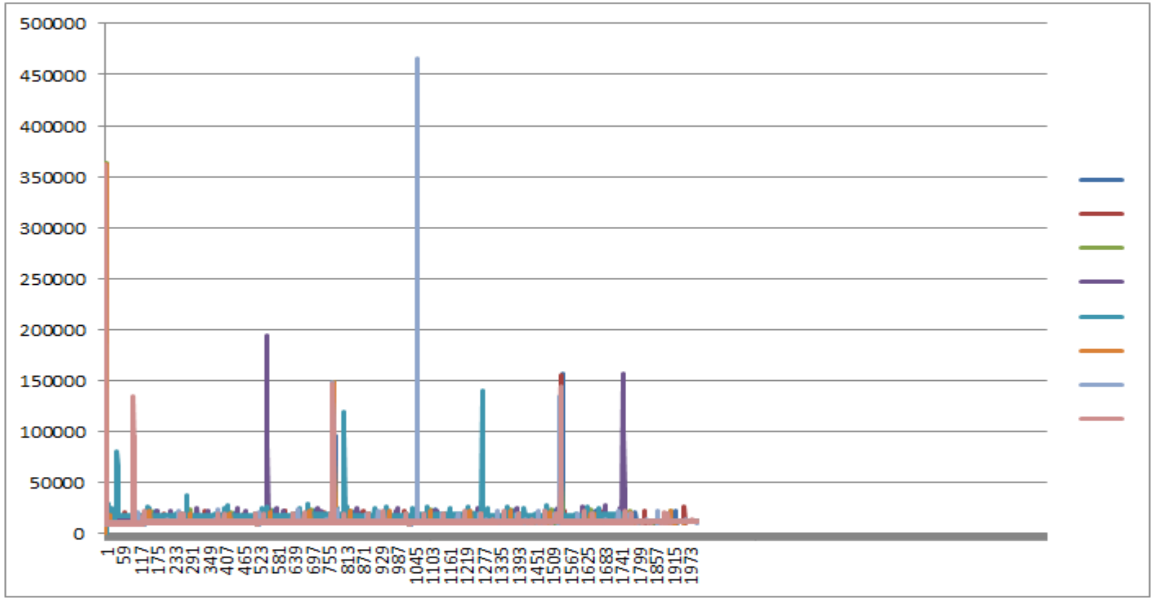


bytes

average time of ~12000 us/byte

Writing blocks to SD card

microseconds

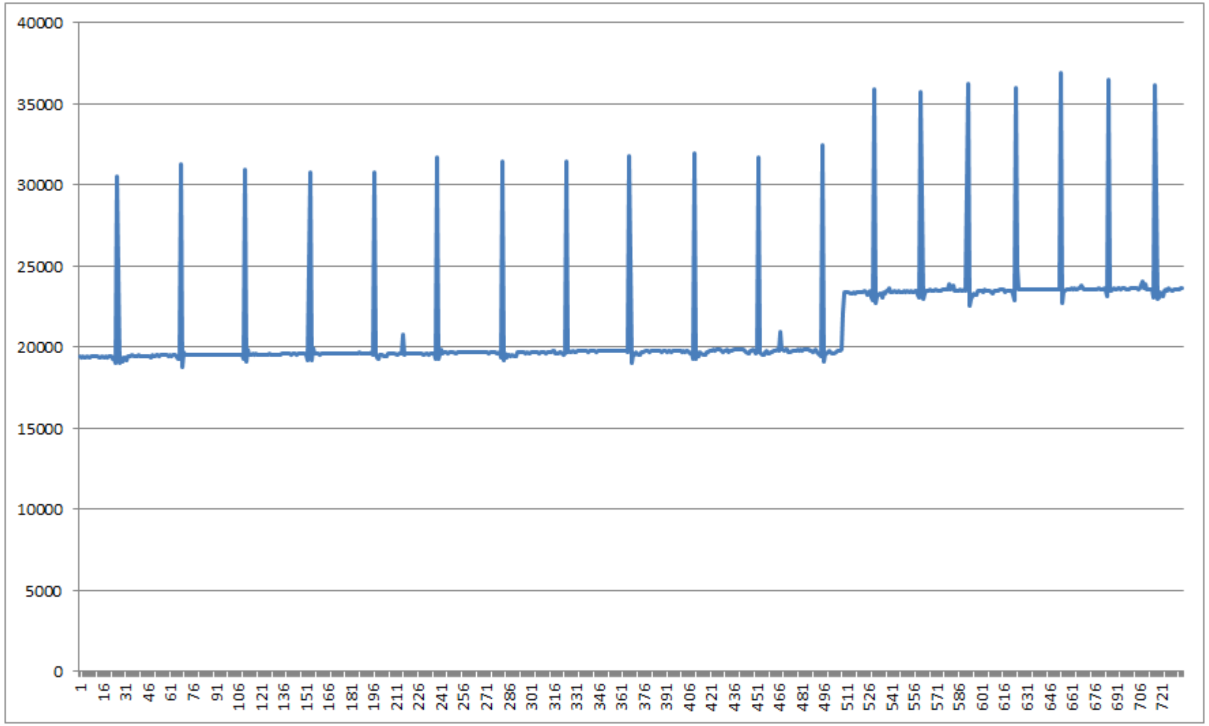


256 byte blocks

spurious spikes due to “erasing large flash blocks and remapping bad spots” - fat16lib (aka some guy online)

pattern breaks down at about 15kB with 100,000 - 400,000 us spikes

SD card write speed tests

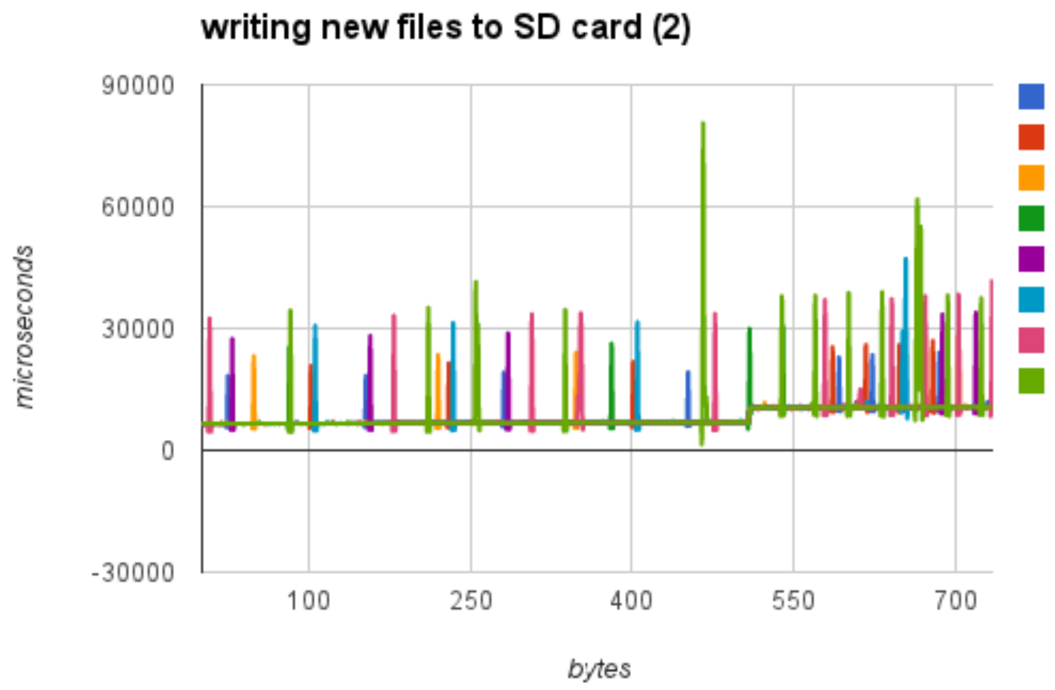


us

size of file (bytes)

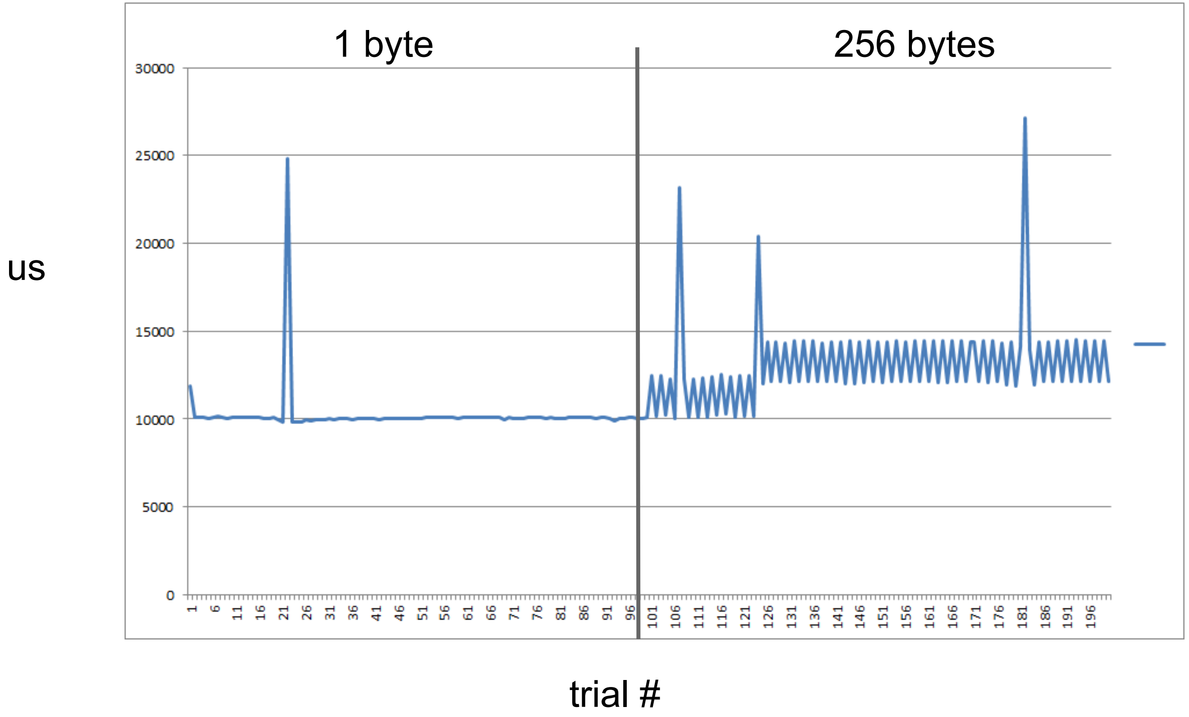
FAT32 filesystem writes 512 byte sector sizes

SD card write speed tests



FAT32 filesystem writes 512 byte sector sizes

First byte penalty



1 byte average = ~12000 us/byte

256 byte average = ~13000 us
or 50 us/byte average!

SD card write strategies

- **Dumb and fast: High speed but inconsistent sampling**
    - use one “large” buffer (<2kB bytes) to take fast burst sample, then write to SD card
    - achieve upper limit of ~10 kHz sampling rate. Might miss data
- **Safe and slow: slower consistent for buffer >~15kB**
    - safe from huge spikes if <~ 640Hz sampling rate
- **Safe and fast: Consistent sampling with buffer <~15kB**
    - safe from spikes if ~8.5kHz max sampling rate

Our solution

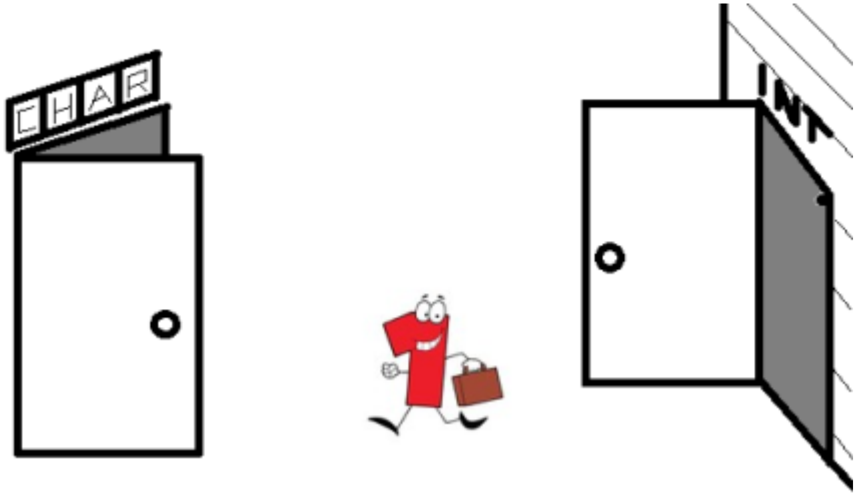
- **Safe and Fast: Consistent sampling with buffer <~15kB**
    - two 256 byte interchangeable buffers; one for writing to SD, one for saving samples
    - worst case write: 30,000us/block or 33 writes/s. -> ~8000 bytes/s
    - Interrupts take samples with period of 8kHz
    - Use mutex to protect buffers from stepping on each other

Data Format

- Arduino can’t write int arrays to SD, but can write strings

- AnalogRead’s max is 1024 so only using 10 bits

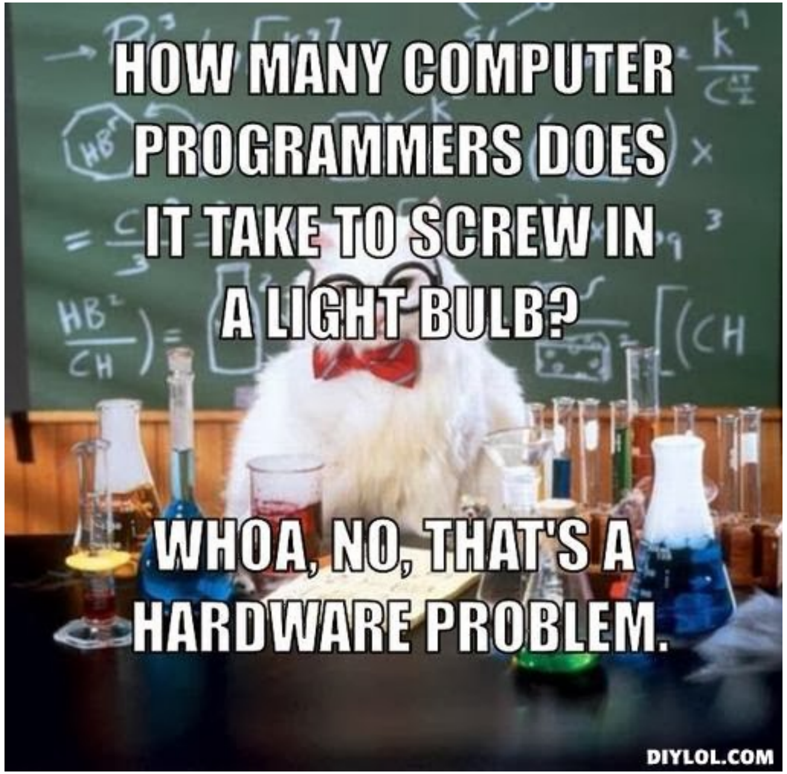
- Gave up two bits of accuracy and saved each value as a char



And does it work?

- Hardware… :-(

- Everything else works


HOW MANY COMPUTER PROGRAMMERS DOES IT TAKE TO SCREW IN A LIGHT BULB?
WHOA, NO, THAT'S A HARDWARE PROBLEM.
DIYLOL.COM

Questions?