Vehicle Data Logging Device

Group 17

Sponsored by
Dr Yiannis Papelis

Kyle Fiducia (EE)
Joshua Mahaz (CpE)
Graham Smith (EE)

Abstract

- Sponsored by Dr. Papelis (UCF CECS)
- Build a compact hardware device that logs data acquired by various sensors on a passenger vehicle
- Vehicle data to be logged includes: geographic location, engine RPM, throttle position, acceleration, speed, yaw rate, and forward looking video
- Device must include some method of configuration as well as an easy way to retrieve the data
- Must allow for widespread data gathering usage
- Device must be portable and easy to install

Motivation

- Dr. Papelis' Motivation:
  - 45,000 automobile accident deaths each year, of which 90% involve driver error.
  - To provide real world data for input into his research.
  - Extensive amounts of logged data to study driver behavior and develop metrics for driver performance.

- Group 17's Motivation:
  - Solid-state device with current standard components
    - Working with standard hardware will be much more versatile than learning a highly specialized task to complete a project.
  - It is also encouraging that if the design is good enough, the device could be actually used after the project is done, rather than just a proof of concept.
  - Sponsored - meaning little money out of pocket.

Vehicle Data Logging Device

- **Data Requirements:**
  - Geographic position
  - Velocity
  - Throttle-position
  - Engine RPM
  - Lateral & longitudinal acceleration
  - Forward looking video images (cost dependent)
  - Yaw rate (optional)
  - Following distance (optional)

- **Administrative Requirements:**
  - Allow for easy data retrieval
  - User configurability
  - User Interface
  - Small, portable, easy to install device
  - Maintain system cost reproduction below ~$400
  - Intelligent Power Supply
    - Draws power only when the engine is running

Project Specifications

	Capture Frequencies	
Data	Minimum	Optimal
OBDII - Speed, RPM, Throttle Position	1 Hz	3 Hz
Lateral & Longitudinal Acceleration	>10 Hz	10 Hz
GPS Position, Speed, Time	1 Hz	1 Hz
Forward Looking Video Images	>1 Hz	2 Hz
Yaw Rate	>5 Hz	10 Hz
Following Distance	>1 Hz	5 Hz

Hardware Requirements

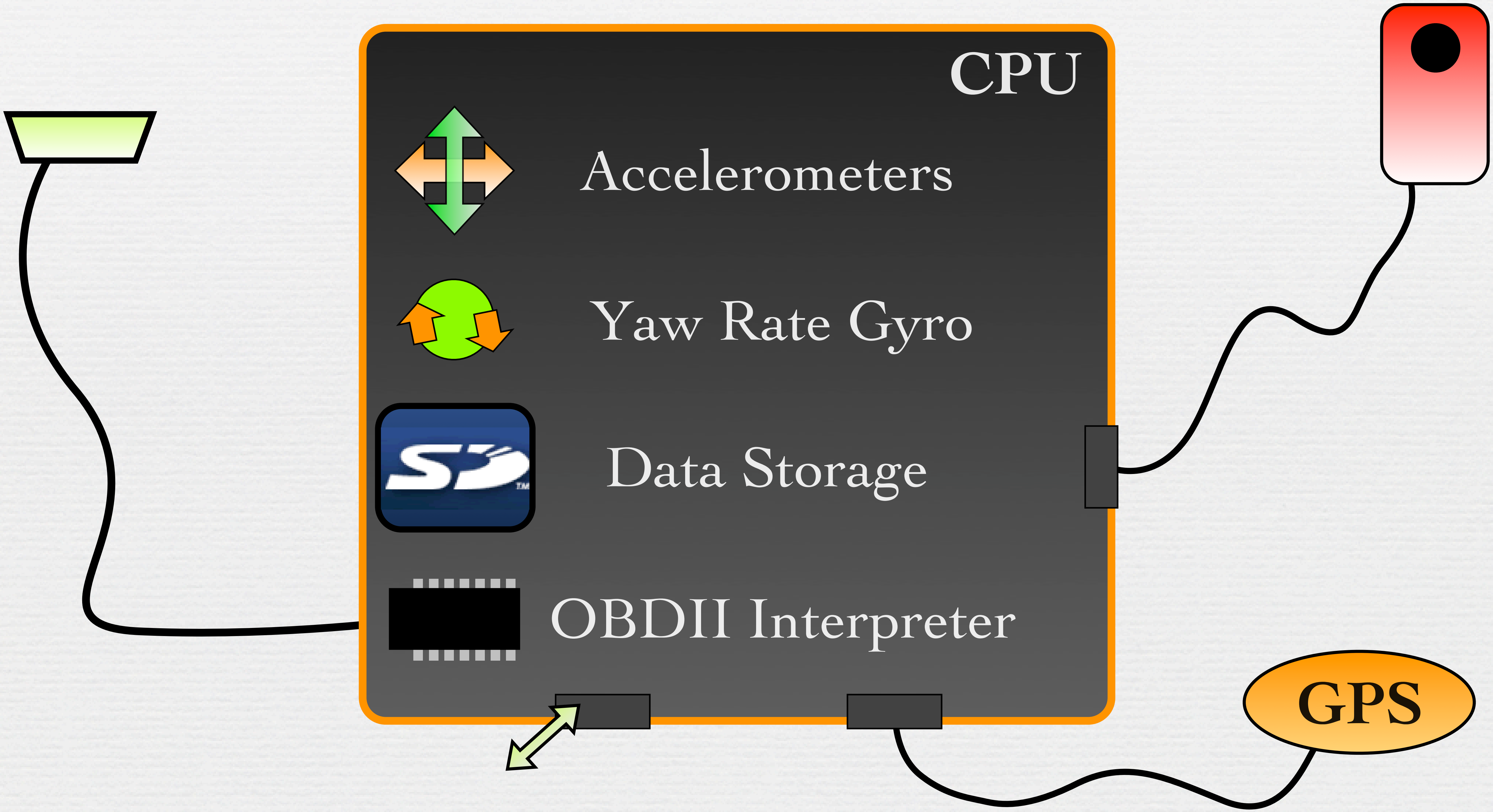
- OBDII Interface (On Board Diagnostics)
  - Vehicle Speed, RPM, Throttle Position (if avail.)
- Accelerometer (dual-axis)
  - Lateral and longitudinal acceleration
- GPS Sensor
  - Vehicle position and accurate time
- Data Storage Device
  - To store all the logged data

Optional:

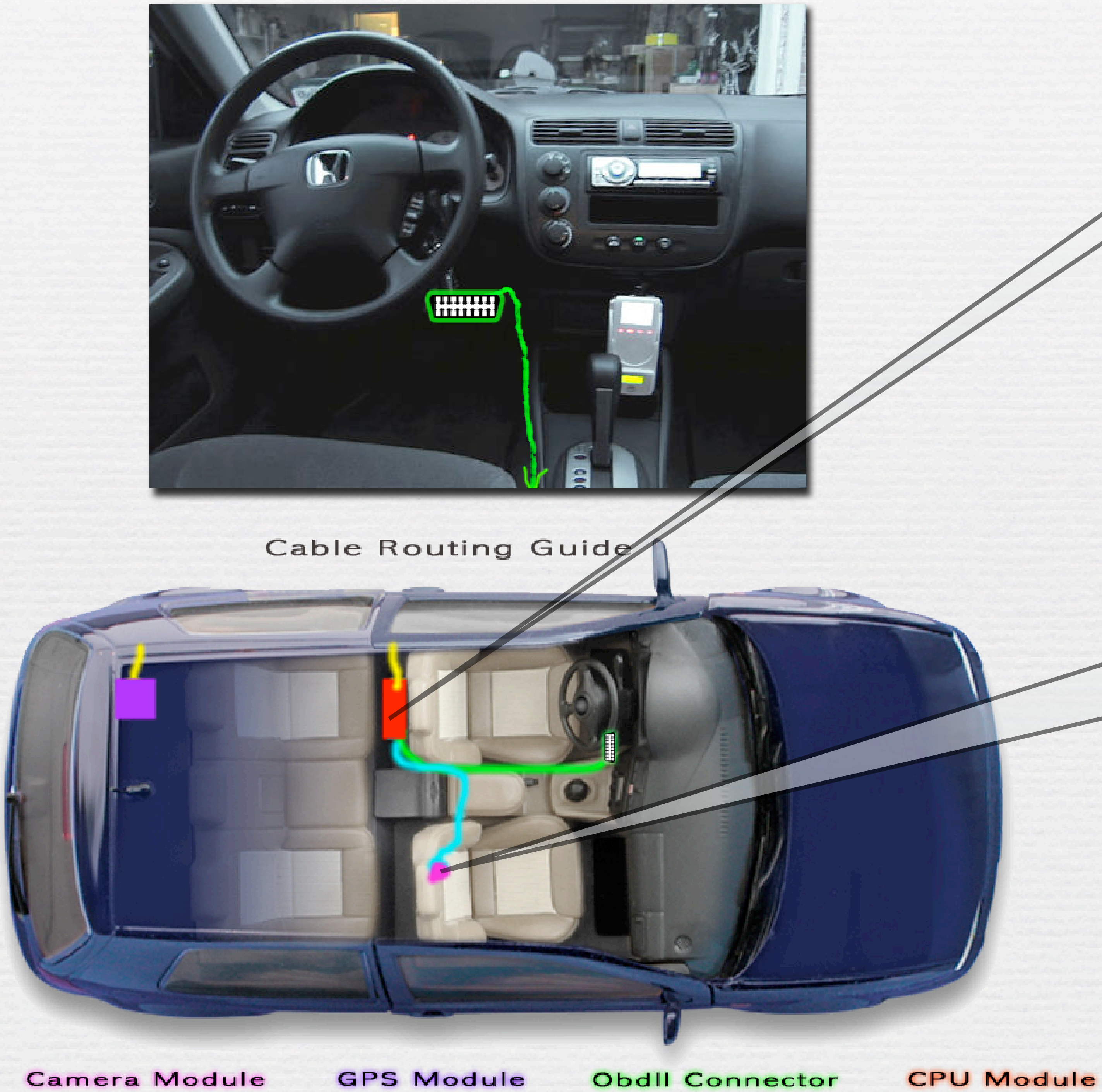
- Image Sensor
  - Captures forward looking video
- Yaw Rate Sensor
  - Measures turning rate of the vehicle
- Range-finder
  - Measures following distance

Hardware Components

Block Diagram



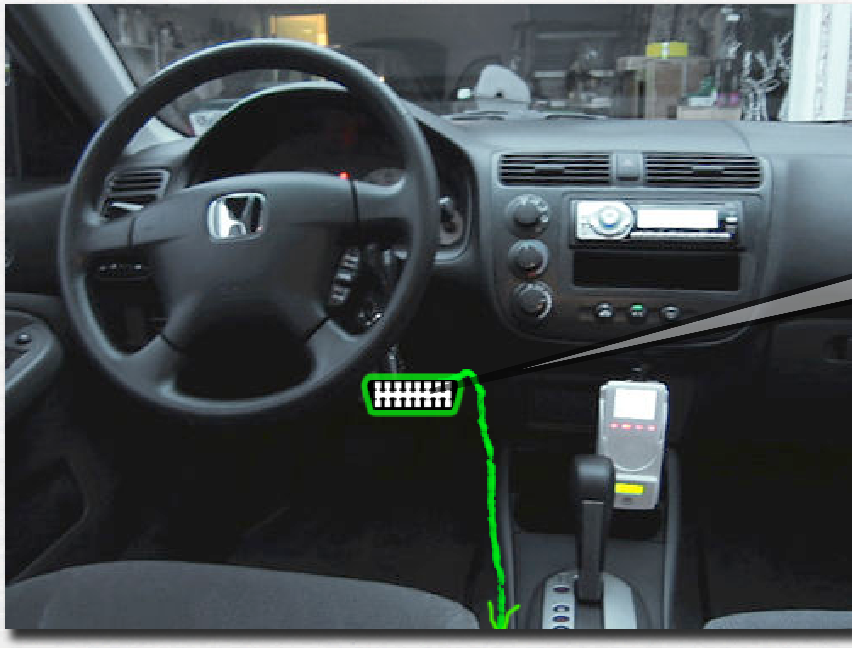
Component Layout



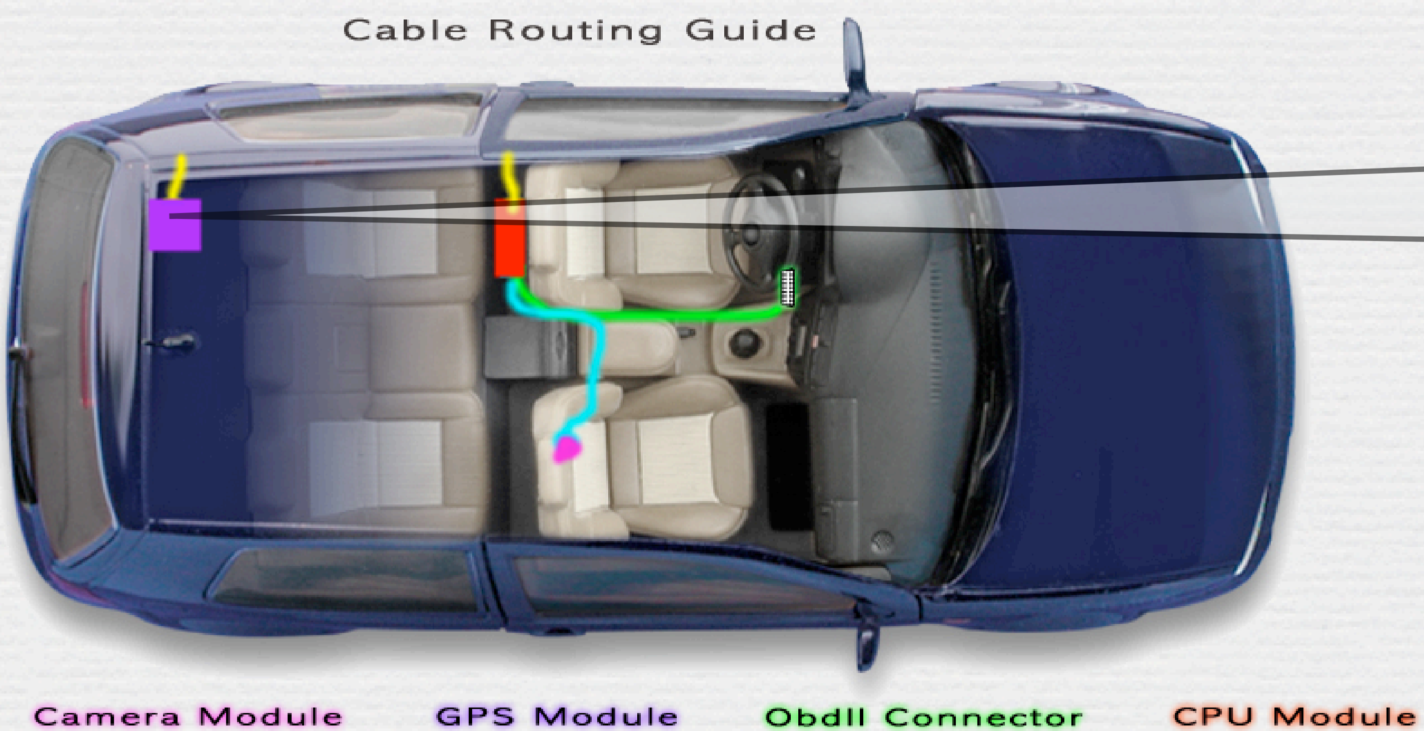
Device CPU placed on the back seat floor

Camera placed in view of driver and road; Attached to headrest of passenger seat

Component Layout

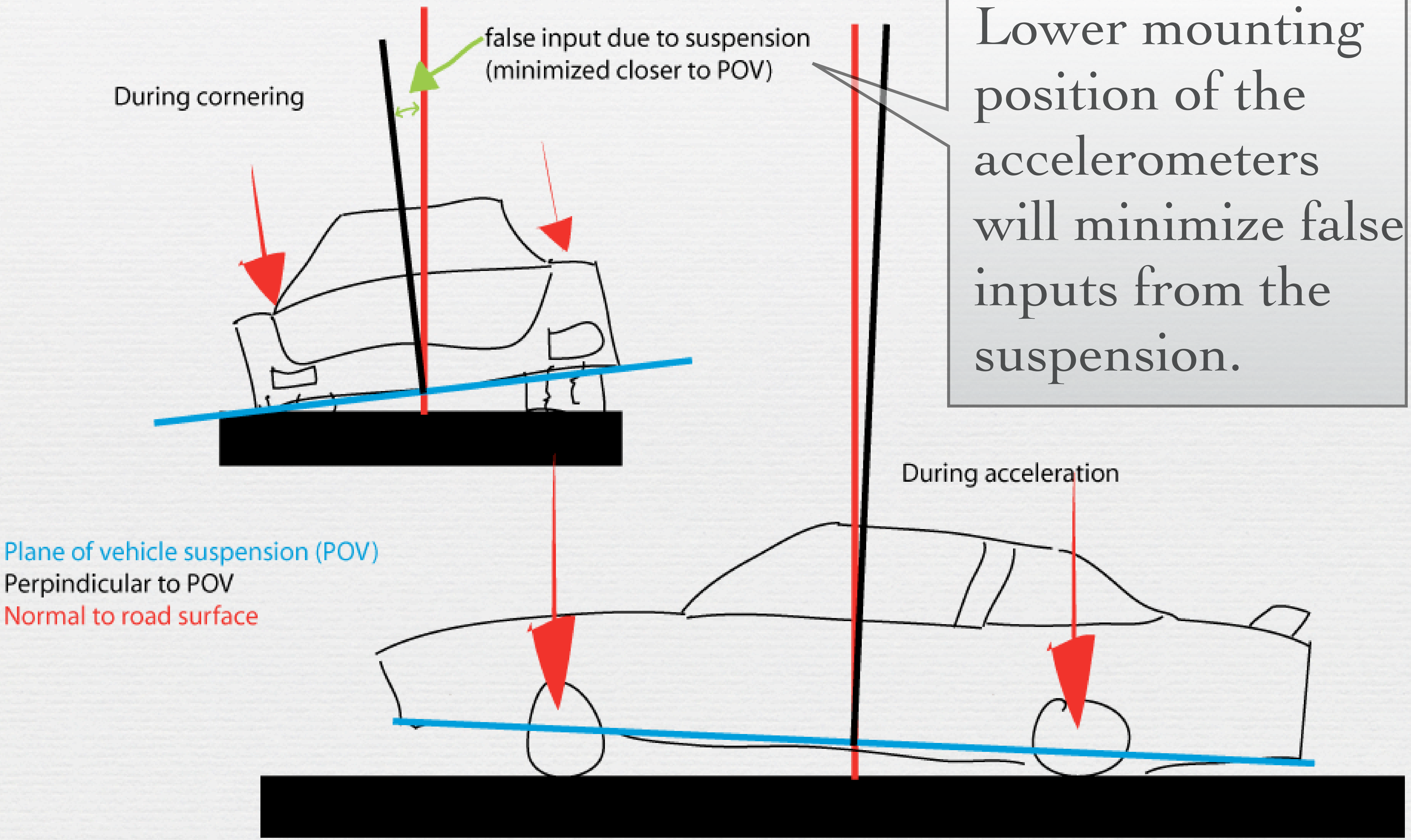


OBDII Standard requires port to be within 36" of steering wheel.



GPS device mounted on roof for clear view of sky and proper ground plane.

Mounting Location Constraints



Lower mounting position of the accelerometers will minimize false inputs from the suspension.

Data Logging Constraints

Image Capture Constraints

Resolution	**Approx File Size (KB)**	**Approx Number of Images/Capacity(MB)**				
Capacities (MB) ---->		128	512	1024	2048	4096
VGA	32.3 KB	4,058	16,232	32,464	64,927	129,855
QVGA	11.0 KB	11,967	47,869	95,739	191,477	382,954
160x120	6.8 KB	19,361	77,443	154,886	309,771	619,543
Text	0.2 KB	613,788	2,455,154	4,910,307	9,820,615	19,641,230
Hours of Image Capture @ 1Hz						
640x480	VGA	1	5	9	18	36
320x240	QVGA	3	13	27	53	106
160x120	sub-QVGA	5	22	43	86	172
Text		170	682	1364	2728	5456

Image Capture Constraints

Resolution	Approx File Size (KB)	Approx Number of Images/Capacity(MB)				
Capacities (MB) ---->		128	512	1024	2048	4096
VGA	32.3 KB	4,058	16,232	32,464	64,927	129,855
QVGA	11.0 KB	11,967	47,869	95,739	191,477	382,954
160x120	6.8 KB	19,361	77,443	154,886	309,771	619,543
Text	0.2 KB	613,788	2,455,154	4,910,307	9,820,615	19,641,230
Hours of Image Capture @ 1Hz						
640x480	VGA	1	5	9	18	36
320x240	QVGA	3	13	27	53	106
160x120	sub-QVGA	5	22	43	86	172
Text		170	682	1364	2728	5456

Storage Media Comparison

Media Type ->	CompactFlash		SmartMedia	MMC	
Varieties	I	II		MMC	RS-MMC
Maximum storage capacity, MB	8000	12000	128	4096	512
Theoretical maximum capacity	137 GB	137 GB		128 GB	
Data read speed, MB/s	40	40	2		
Data write speed, MB/s	40	40			
Read/write cycles	∞	∞	1,000,000	1,000,000	∞

Media Type ->	Sony Memory Stick (proprietary - not much hardware avail)			
Varieties	Standard	Pro	Pro Duo	Micro
Maximum storage capacity, MB	128	4096	4096	
Theoretical maximum capacity	128 MB	32 GB		
Data read speed, MB/s	2.5	20	20	20
Data write speed, MB/s	1.8	1.8	10	
Read/write cycles	∞	∞	∞	∞

Benefits of SD

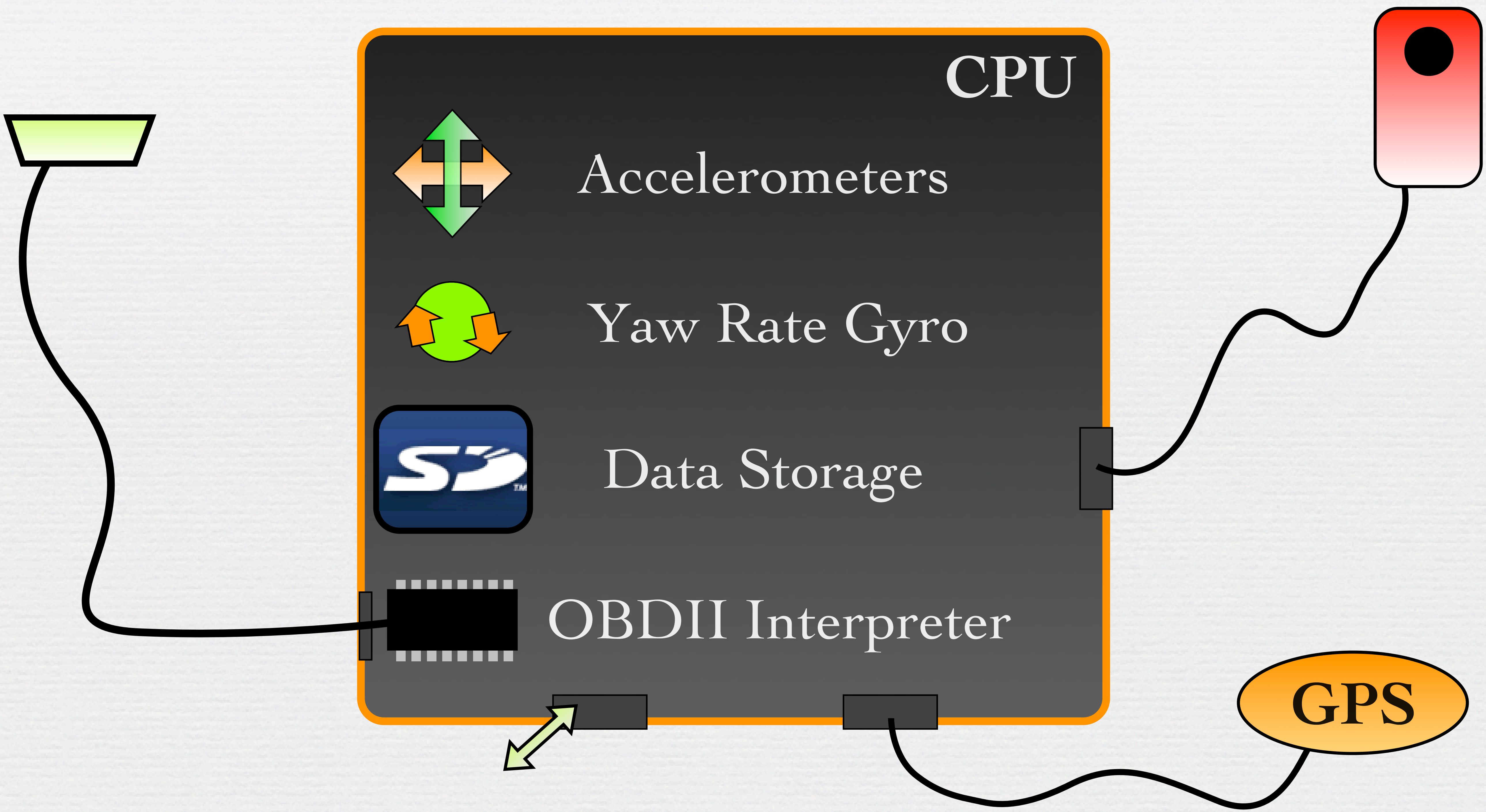
- Uses less power than CF
- Lots of 3rd party hardware available
- Smaller
- Inexpensive cards in many sizes

Media Type ->	xD			Secure Digital (SD)		
Varieties		Type M	Type H	SD	miniSD	microSD
Maximum storage capacity, MB	512	2048	1024	4096	2048	2048
Theoretical maximum capacity	512 MB	8 GB	8 GB	128 GB		
Data read speed, Mb/s	5	4	15	20		
Data write speed, Mb/s	3	2.5	9	20		
Read/write cycles	∞	∞	∞	∞		

Hardware

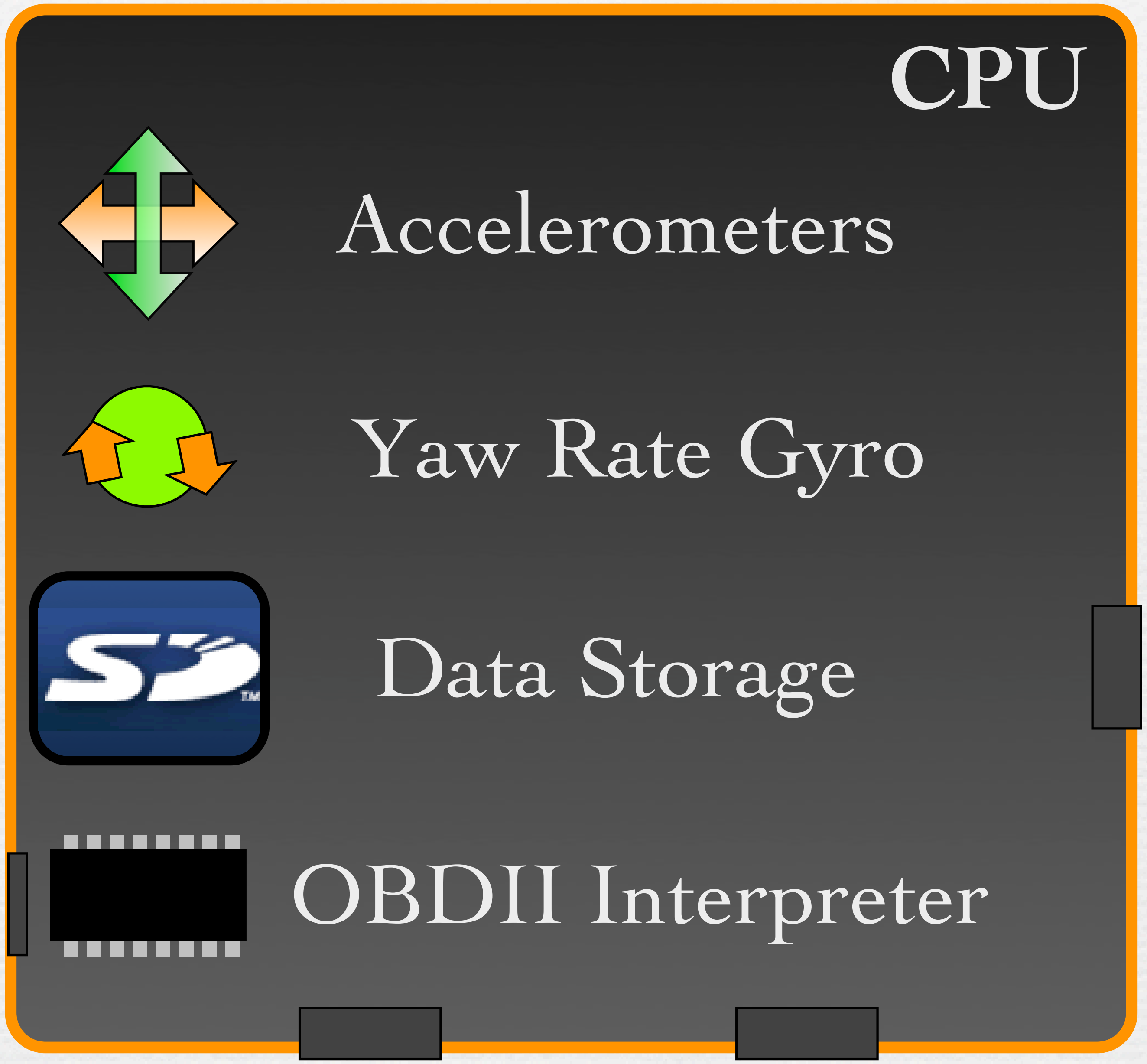
Hardware Components

Block Diagram

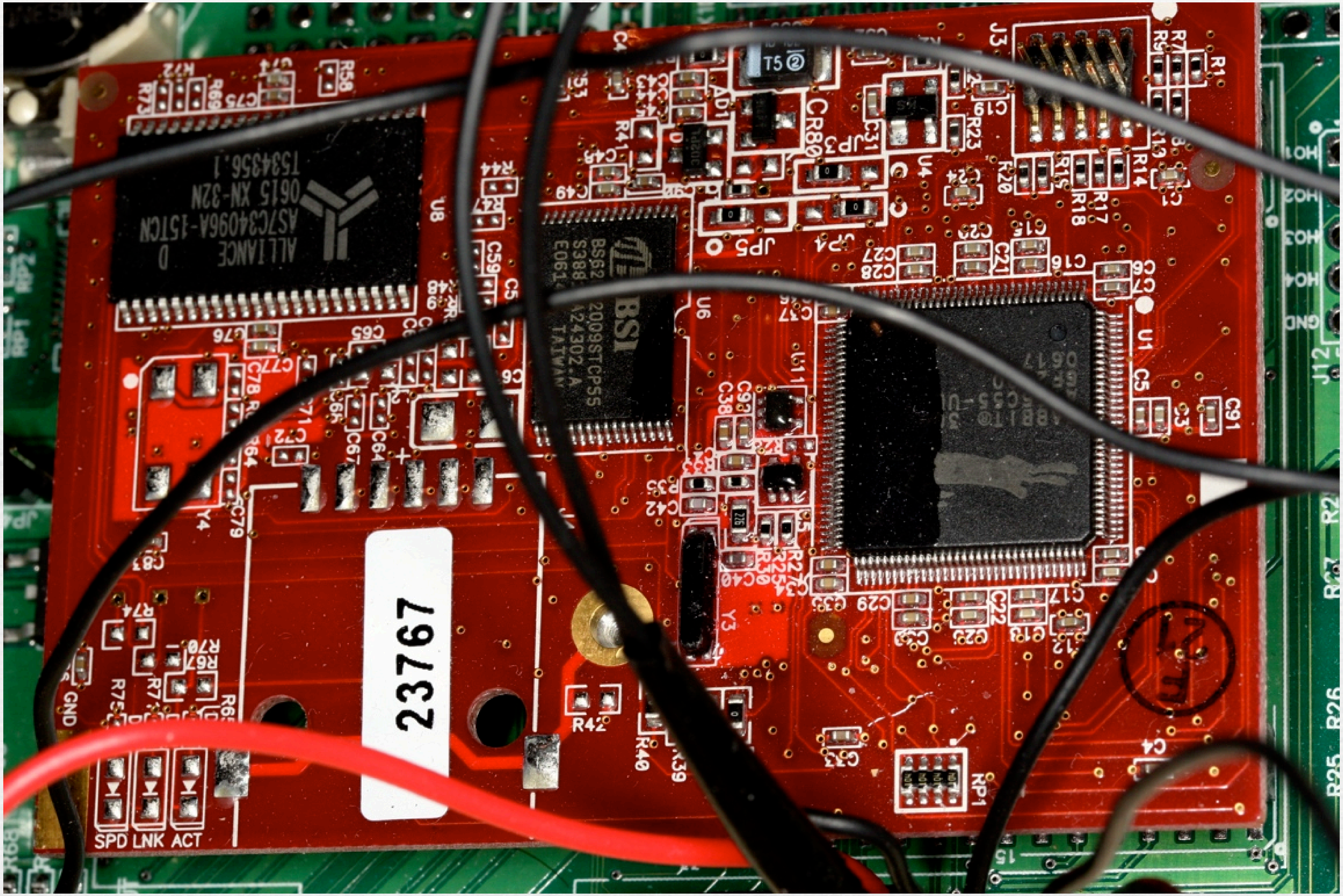


Hardware Components

Block Diagram

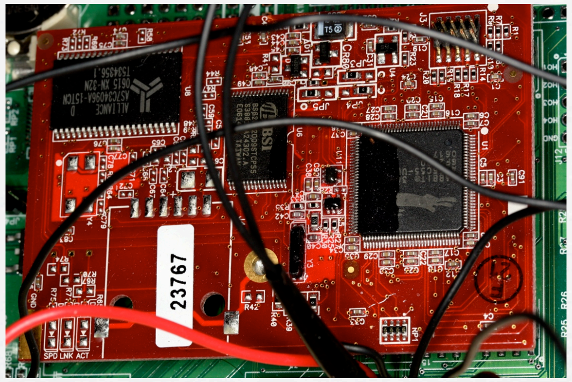


Micro-Controller



Vehicle Data Logging Device

Micro-Controller



Rabbit 3220

Design Considerations

- **Speed - 44MHz**
  - Adequate speed for data throughput
    *Image capture is largest constraint
- **I/O Lines**
  - Must have adequate I/O lines to support all devices (33)
    Rabbit has 52 I/O Pins
- **Protocol Support**
  - I^2C, RS232, SPI, PWM Input Capture
- **Programming Language**
  - Assembly, Dynamic C
- **Cost**
  - Rabbit is actually a bit expensive, $80.00, but Dr Papelis wanted to use it.
  - PIC is a viable option at $10/unit, we are simultaneously developing a PIC version as well.

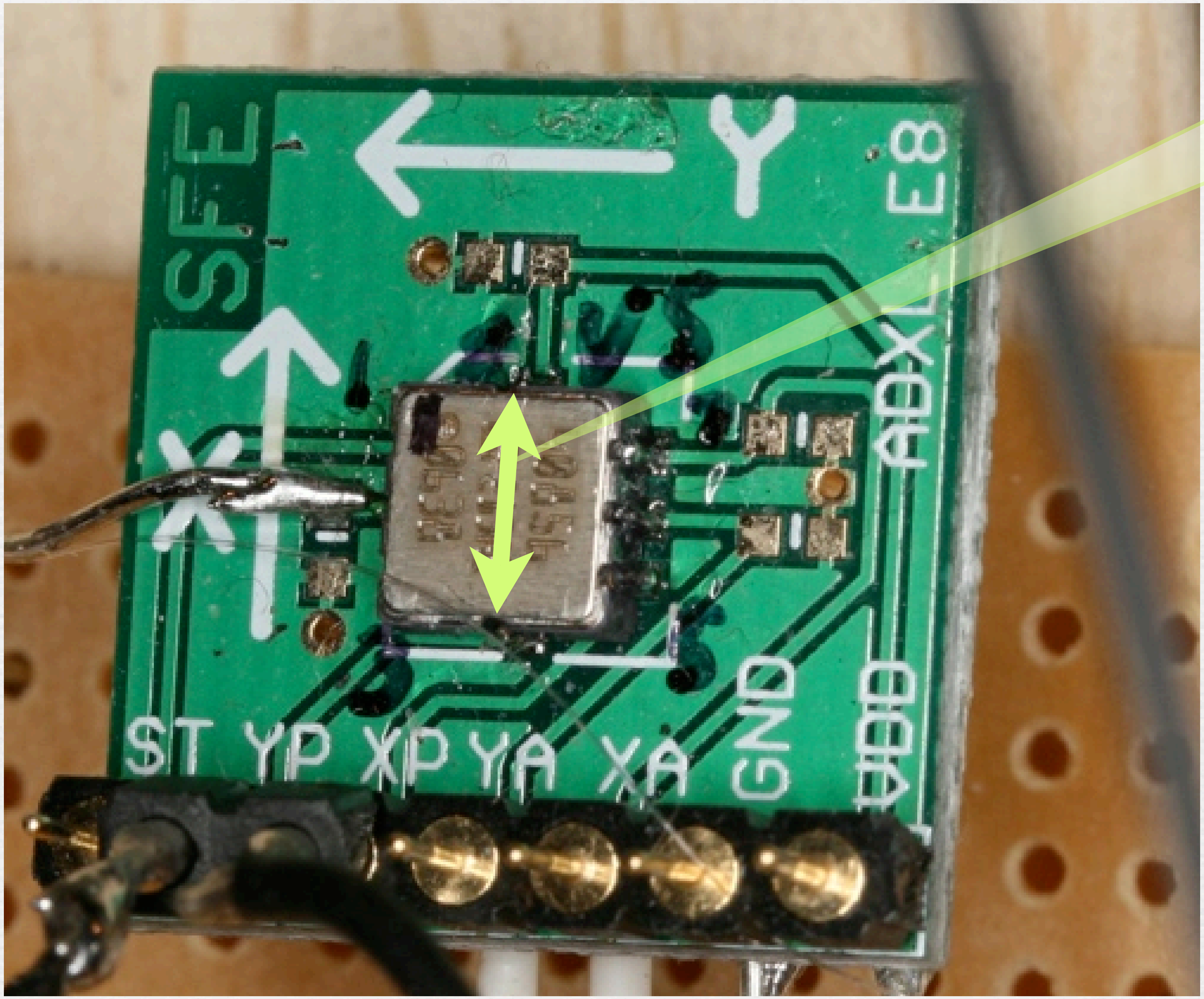
Input/Output Lines

Device	Communication	Data Pins	Other	Total Req
GPS Device	Serial Data	2	0	2
Camera	UART	2	0	2
2-Axis Accelerometer	PWM	2	0	2
Yaw Rate Gyro	SPI	3	1	4
ELM327	Serial	2	2 (Busy/RTS)	4
Diagnostics Port	Serial	2	0	2
Future Expansion	Serial	4	2	6
Input Total				22
uALFAT	UART/SPI	6	0	6
UI LEDs (5)	PWM	0	5	5
Output Total				11
Total				33

Accelerometer

Lateral & Longitudinal Accelerations

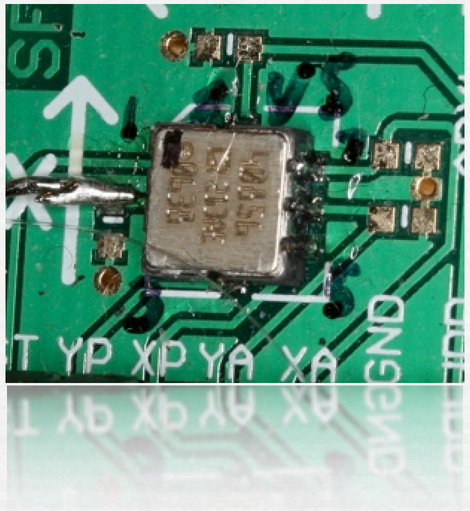
ADXL213



5mm

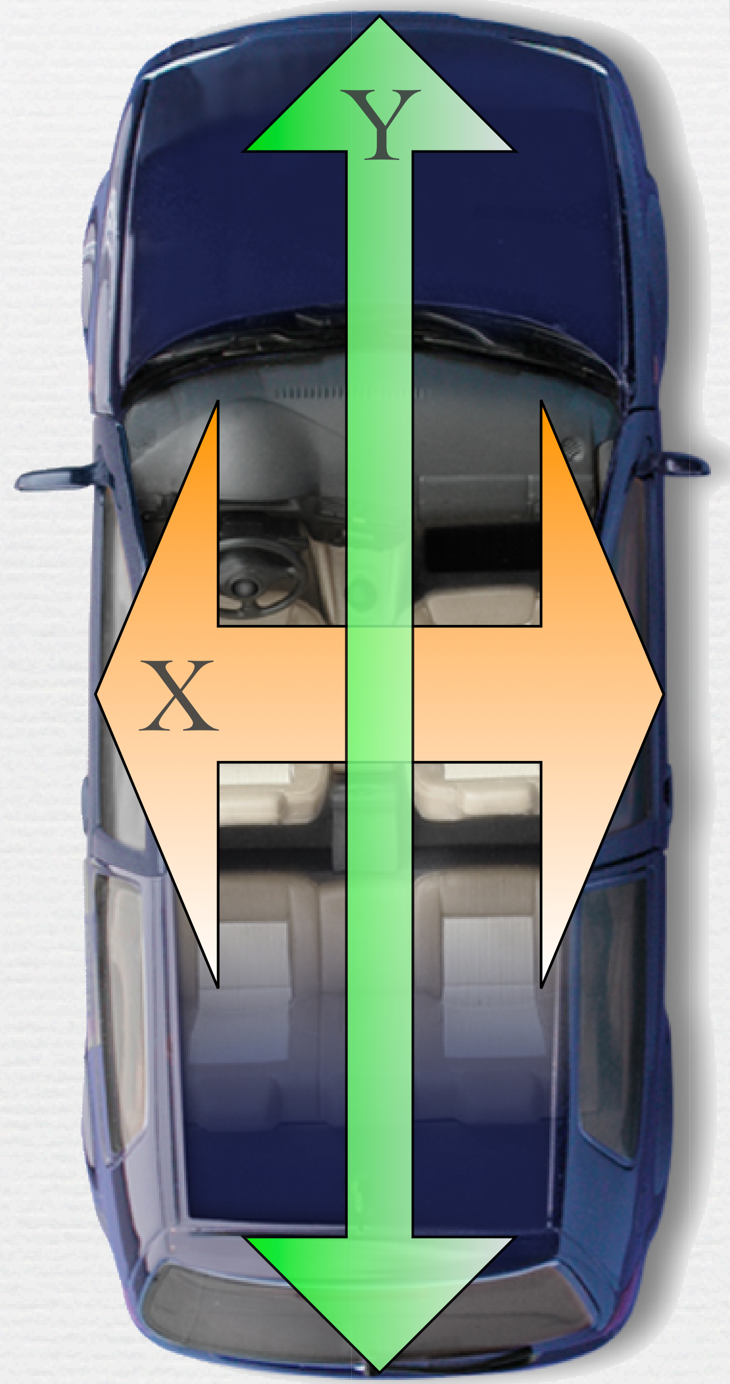
Accelerometer

Lateral & Longitudinal Accelerations



ADXL213

- <u>Design Considerations</u>
  - **Dual axis**
  - **PWM output**
    - Built-in ADC
  - **Range**
    - ±1.2g
  - **Cost**
    - $40.00

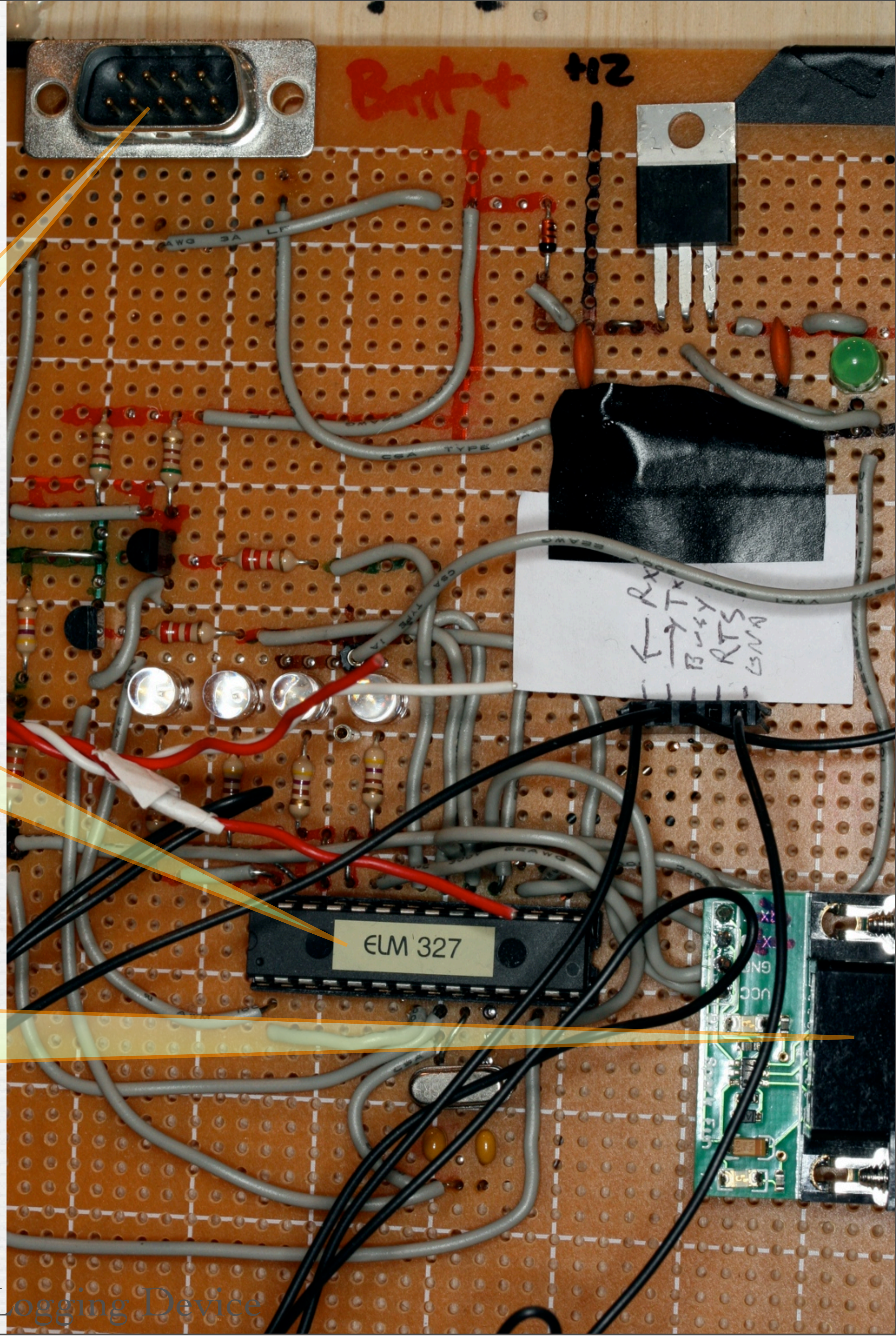


OBDII Interface

To car's OBDII port

OBDII Interpreter

Data to µController



OBDII Interface

Vehicle RPM & Throttle Position

Design Considerations

- **Compatibility**
- **Ease of Use**
  - Uses standard AT command syntax
  - RS232
  - Handles all the bus initiation and detection itself automatically
- **Cost**
  - $30.

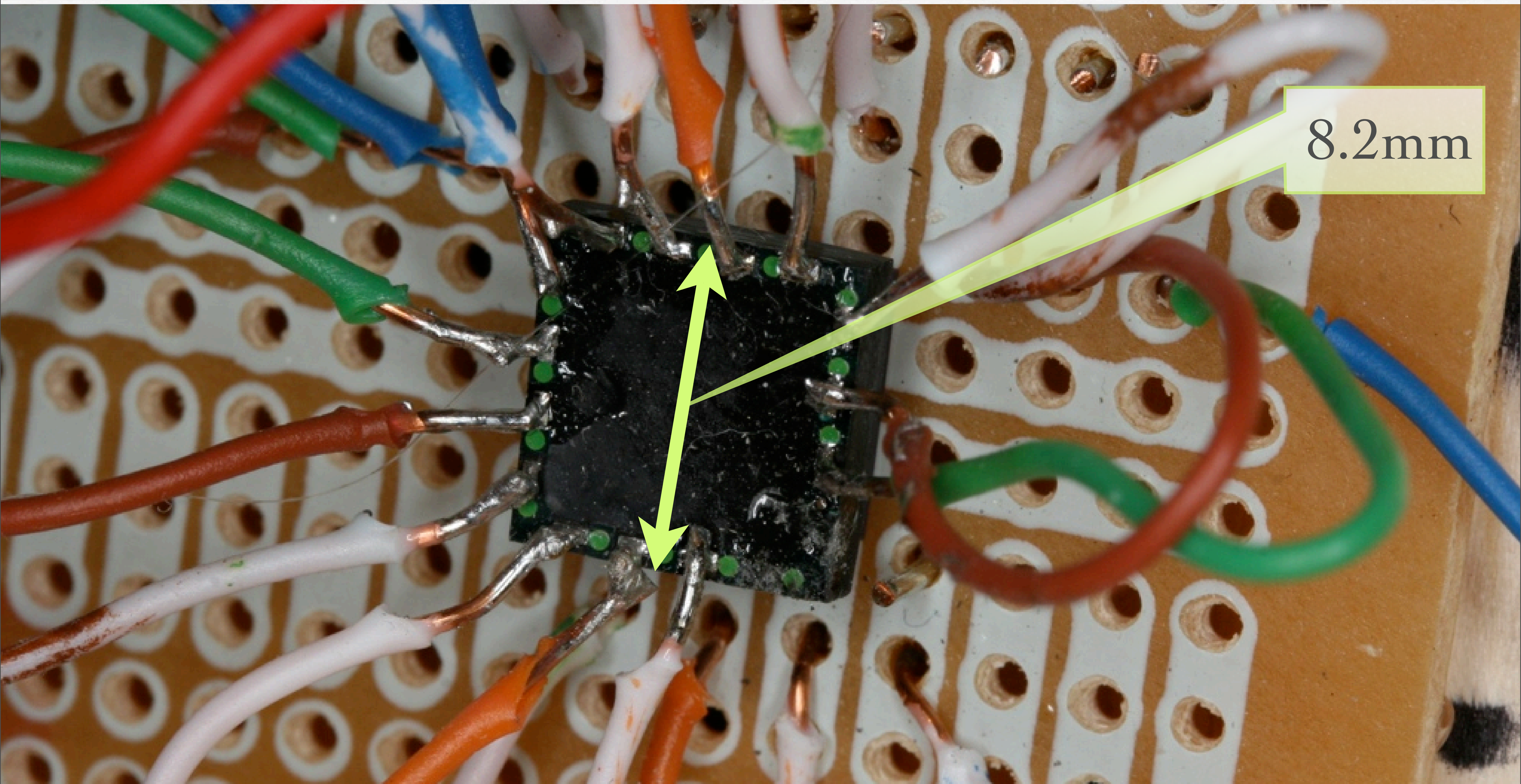
ELM327

Compatible with all OBDII standard protocols:

SAE J1850 PWM	Ford
SAE J1850 VPW	GM
ISO 9141-2	Chrysler, European, Asian
ISO 15765 CAN	all after 2008
ISO 14230	

Yaw-Rate Gyro

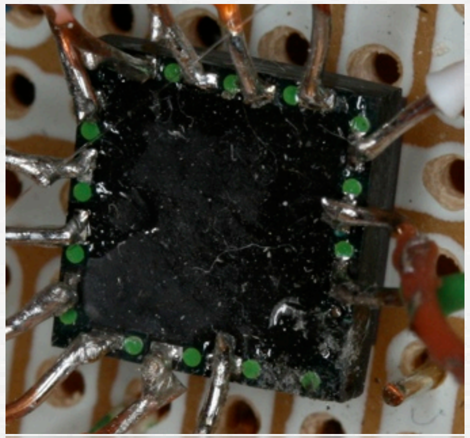
ADIS16100



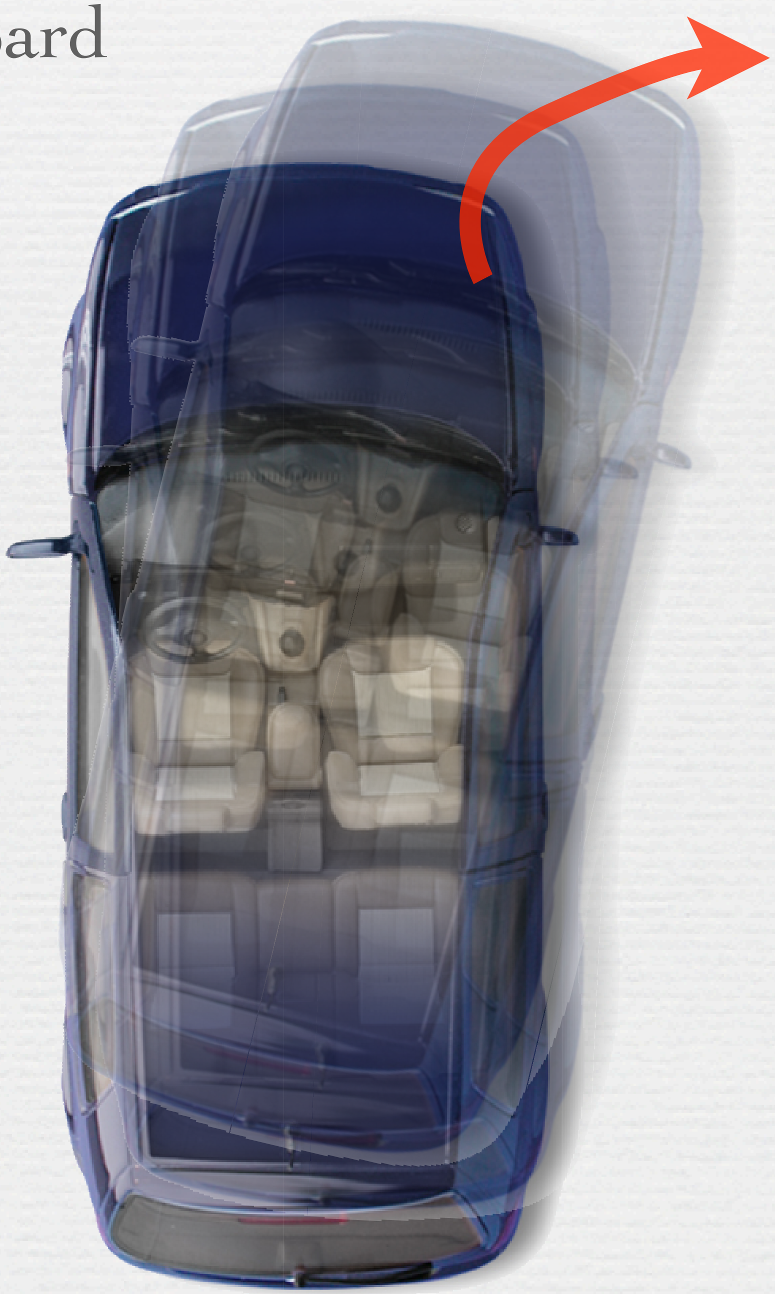
Yaw-Rate Gyro

Vehicle Turning Rate

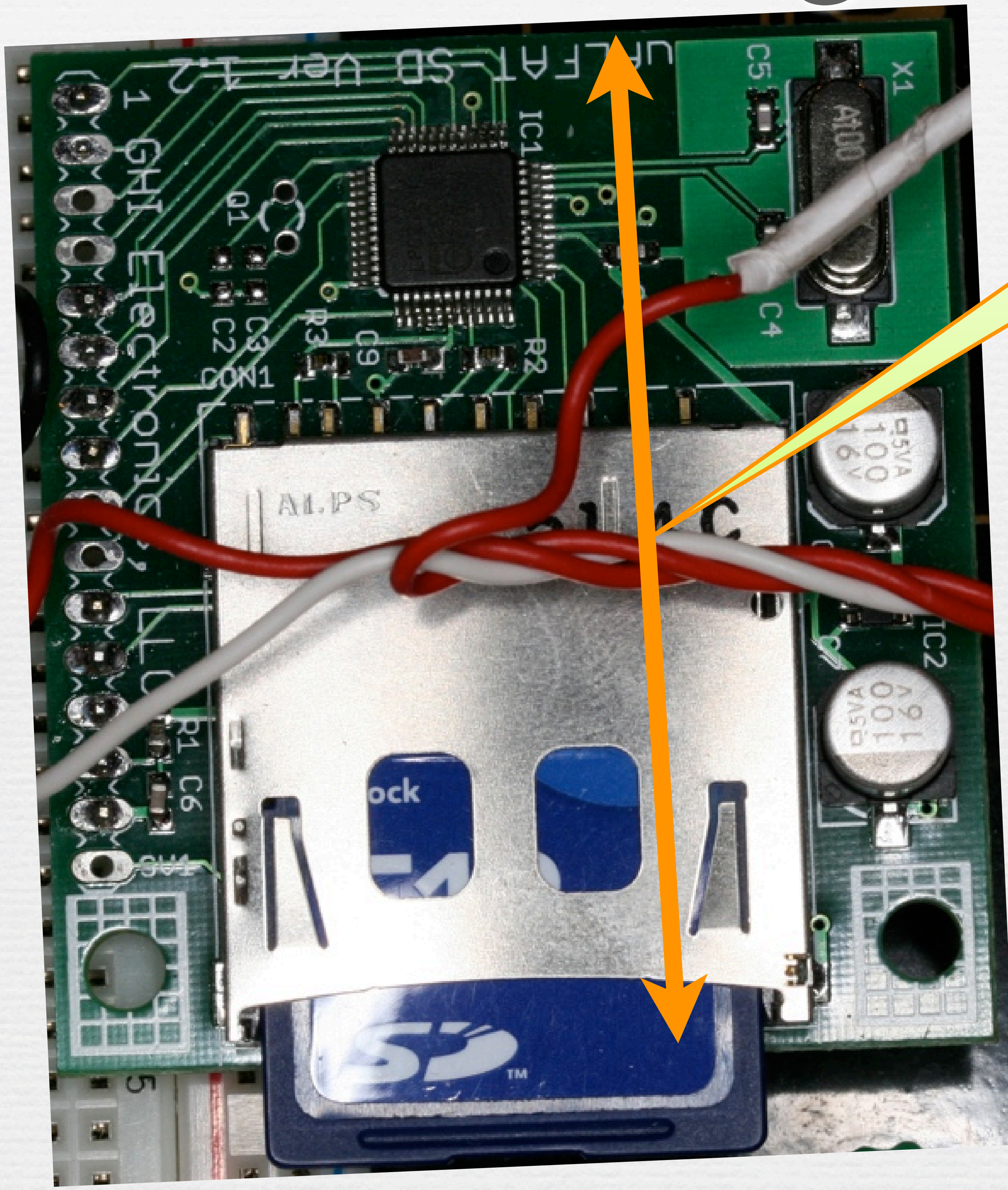
ADIS16100



- Design Considerations
  - **Built-in ADC**
    - Reduces number of components on the board
  - **Resolution**
    - 360°/second
  - **Protocol Support**
    - SPI Interface is fast and relatively easy
  - **Number of Axes**
    - Single axis measurement (yaw)
  - **Cost**
    - A bit expensive at $50 each
  - **Difficulties**
    - LGA16



Data Storage Device



51mm

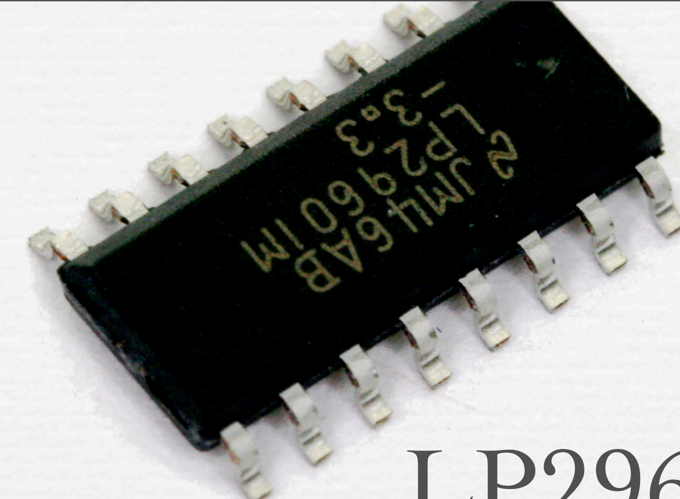
µALFAT

Data Storage Device

µALFAT

- Design Considerations
  - **File System Format**
    - FAT16
  - **Speed**
    - 57.5 KB/s write speed
  - **Physical Media Format**
    - Removable SD Card
  - **Communication**
    - RS232
  - **Cost**
    - $40.00

Power Supply

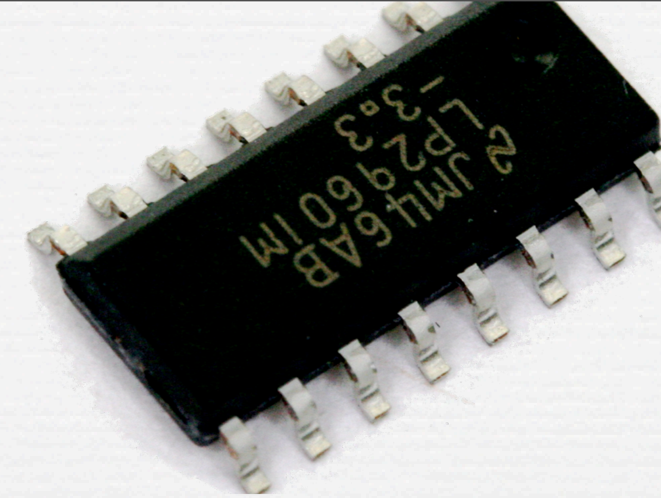


LP2960

Design Considerations

- **Input vs Output Voltages**
  - 11-15 VDC from OBDII port must be converted to a stable 3.3 and 5 VDC for our components.
- **Logic Level Shutdown**
  - Will shut down PSU and attached devices with one pin
- **Status Flag Pin**
  - Indicates status of device
- **Cost**
  - $3.50
- **Internally Fused**
  - 500mA internally limited current
- **Automatic Thermal Limiting**
  - Will shut down before heat damages components

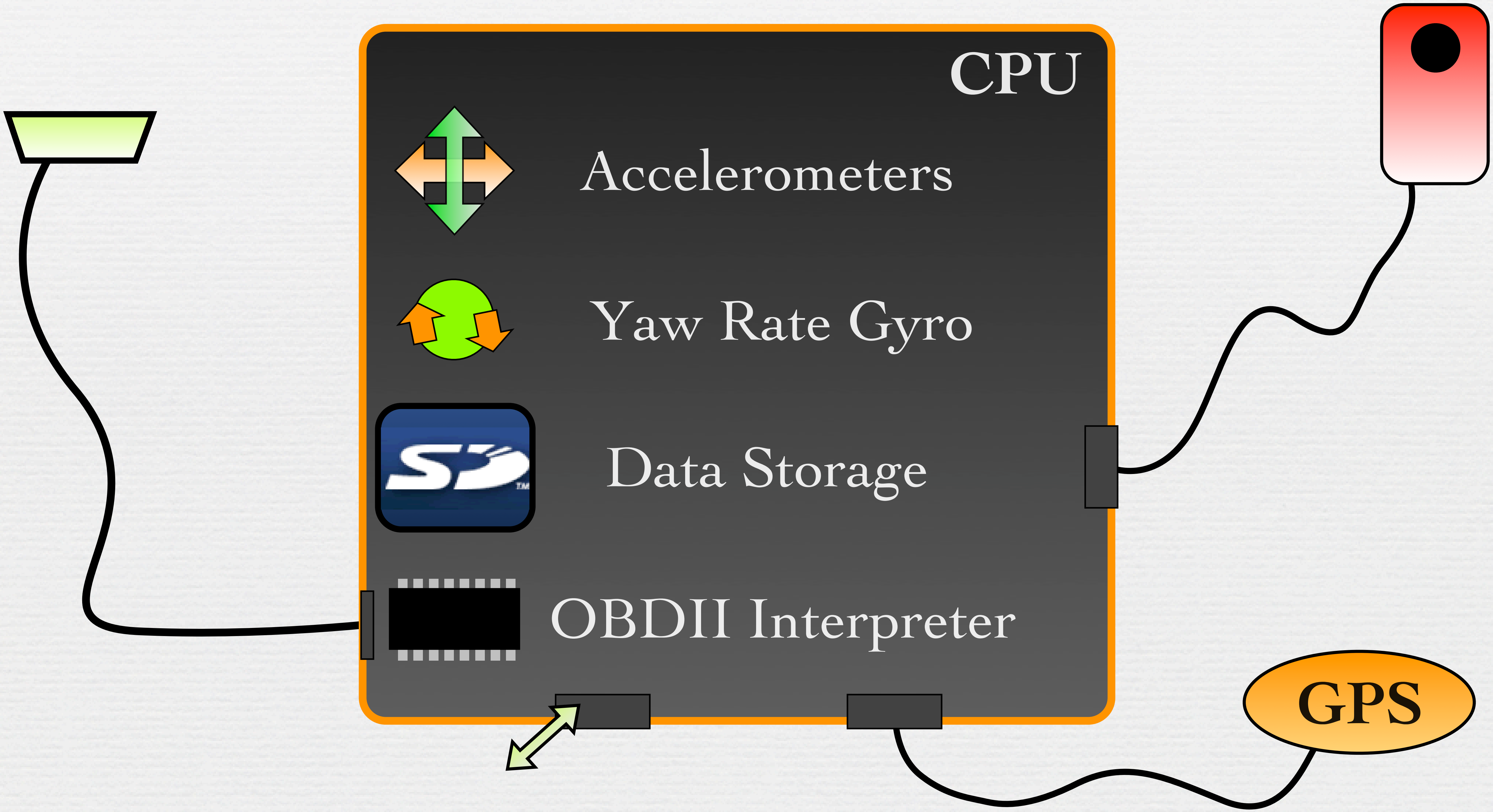
Power Requirements



Part	Voltage (V)	Current (mA)	Power (mW)
Micro-Controller	3.3	255.0	841.5
Storage	3.3	5.0	16.5
Camera	3.3	15.0	49.5
Accelerometers	5.0	0.7	3.5
OBDII Chip	5.0	9.0	45.0
GPS	5.0	60.0	300.0
Yaw Rate Gyro	5.0	7.0	35.0
Total			1291

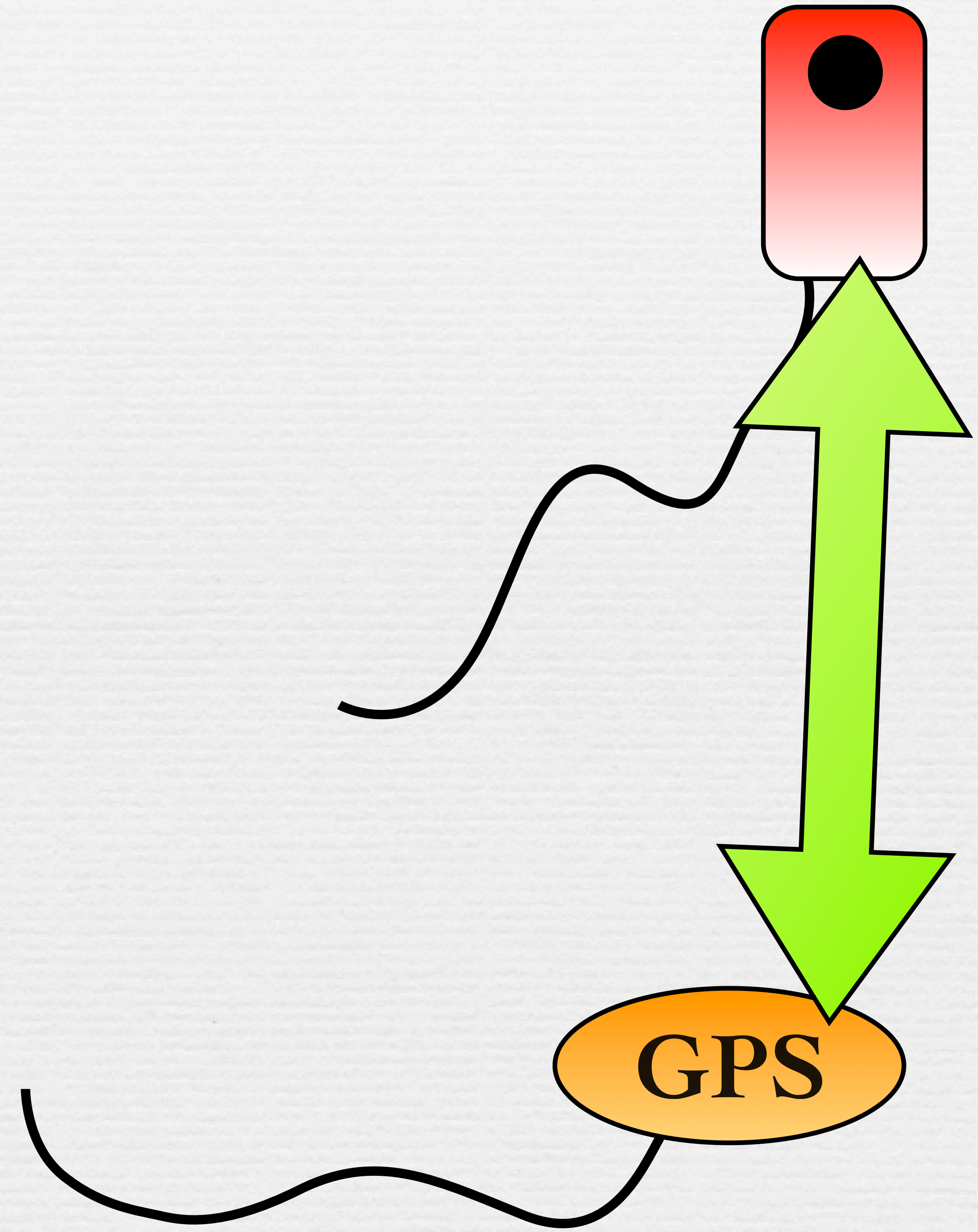
Hardware Components

Block Diagram



Hardware Components

Block Diagram



GPS Sensor

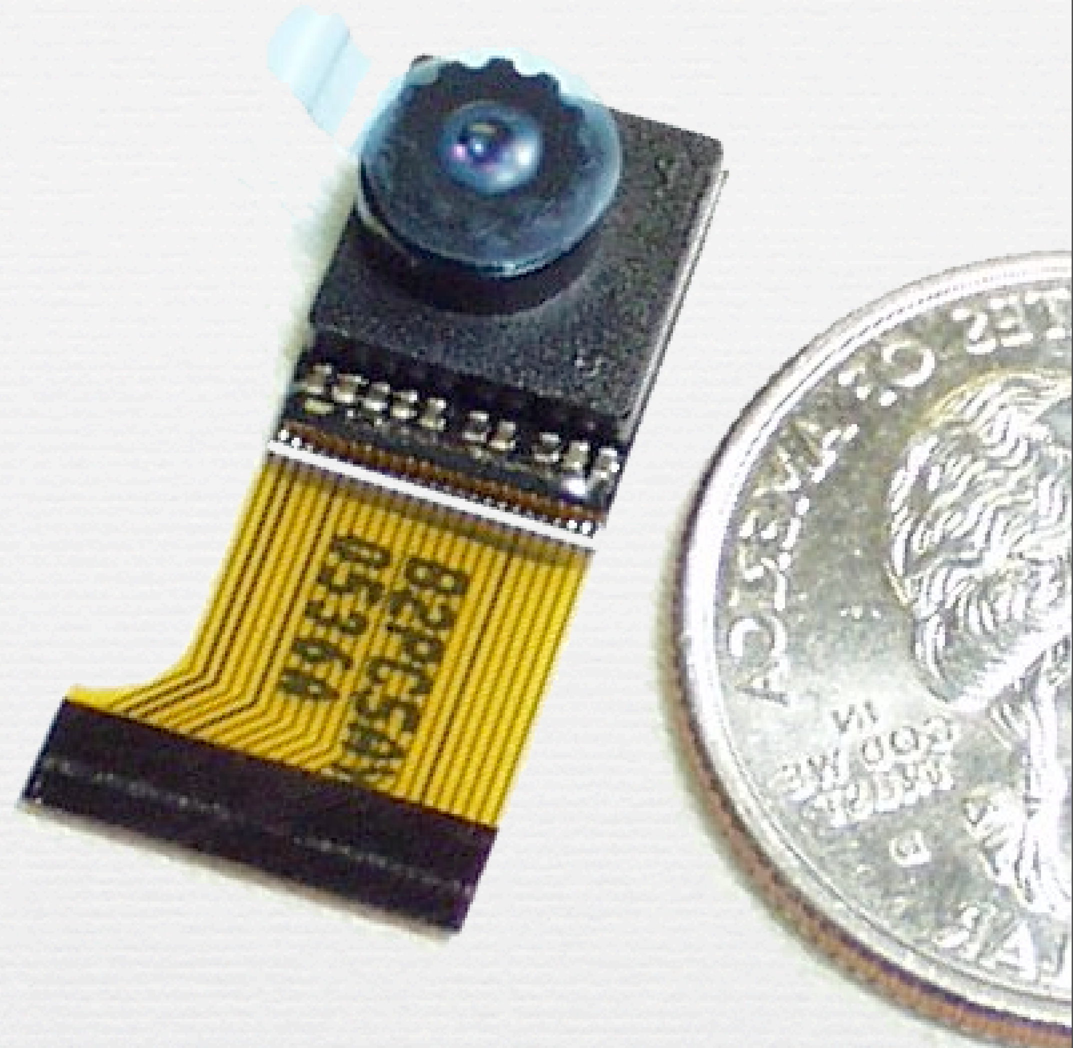
Geographic Position & Speed

Garmin GPS18

Design Considerations

- **Update Frequency**
  - 1 Hz
- **Communication**
  - Standard NMEA over RS232
- **Time To First Fix**
  - ~40 seconds (cold start)
- **Physical Design**
  - Weatherproof, magnetic enclosure
- **Cost**
  - Already had one; $80 to buy

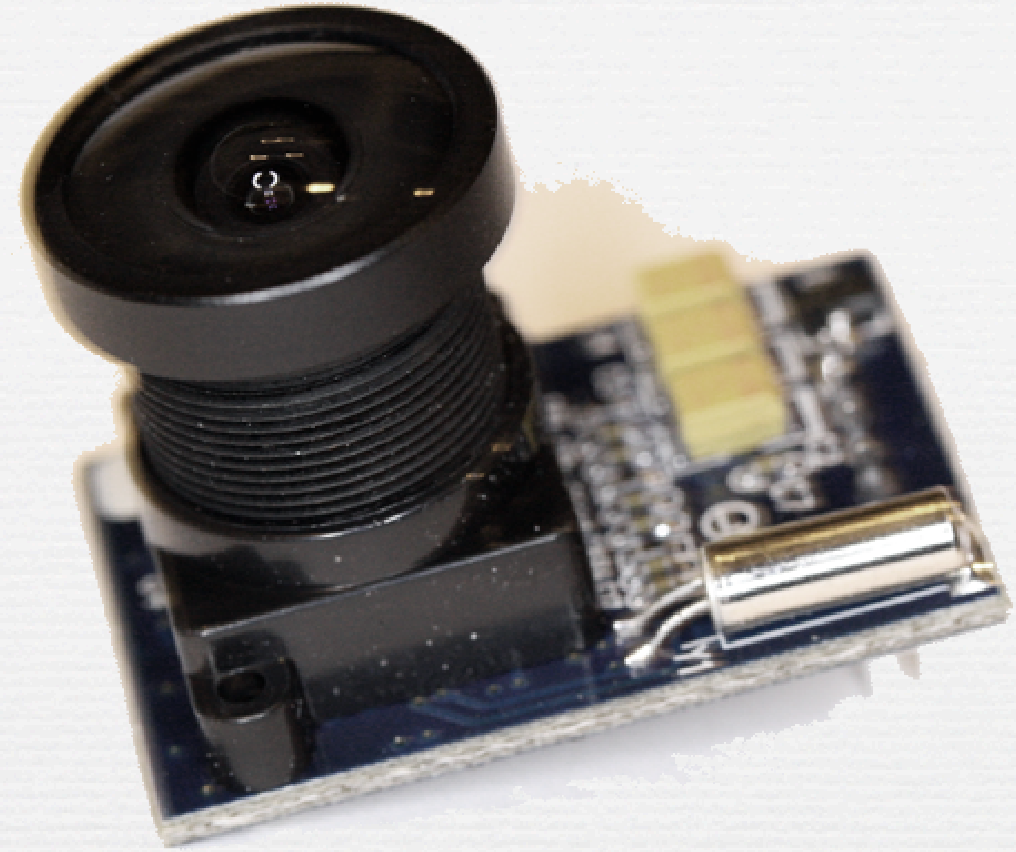
Camera / Video Capture



Design Considerations

- **Resolution**
  - VGA, QVGA, CIF, QCIF
  - Must scale to lower resolutions to enable higher frame rates with the same throughput.
  - Configurable on-the-fly
- **Frame Rate**
  - Only limited by communications speed
- **Video Format/Compression**
  - JPEG output
- **Communications**
  - RS232 @ 115.2Kb/s max
- **Cost**
  - $50./module
- **Wide Field of View**
  - **118°**

Camera / Video Capture



C328-2225BW

Design Considerations

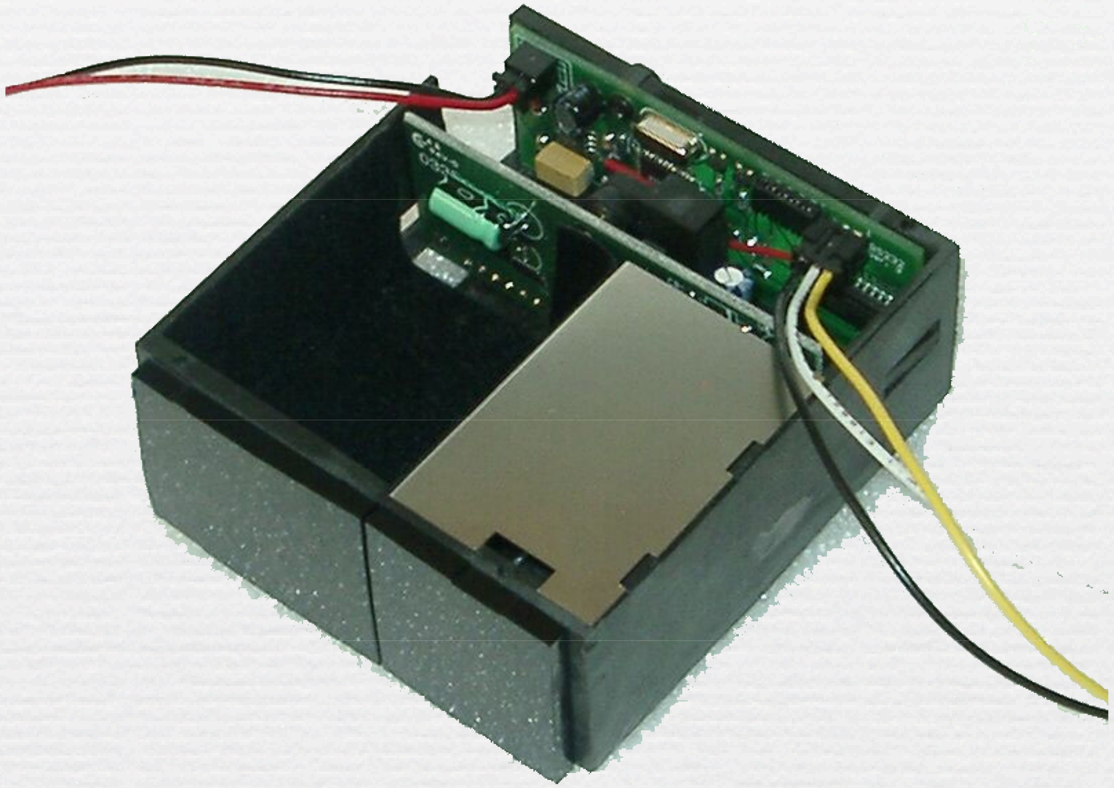
- **Resolution**
  - VGA, QVGA, CIF, QCIF
  - Must scale to lower resolutions to enable higher frame rates with the same throughput.
  - Configurable on-the-fly
- **Frame Rate**
  - Only limited by communications speed
- **Video Format/Compression**
  - JPEG output
- **Communications**
  - RS232 @ 115.2Kb/s max
- **Cost**
  - $50./module
- **Wide Field of View**
  - **118°**

Range Finder

Following Distance

Design Considerations

- **Cost**
  - Laser - $600
  - Microwave - $10,000
  - Budget still only $400
- **FCC Regulations**



Final Product Package

- Final design will be implemented on a custom fabricated printed circuit board (PCB)
- PCB is fabricated so the optional components can be added to the board or not.
  - Allows for lower cost devices with reduced functionality
- The software will determine whether or not the sensors are present and log data accordingly.
  - Intelligent software control to reduce user interaction
- Enclosure design to be finalized when final PCB measurements and layout are available

Software

Software Design

- Sensor Boot Sequences w/ Error Handling
- User Interface & User Configurability
- Polling Multiple Sensors
- Store to Removable Media Device

Sensor Boot Sequence

- Error handling
- Call initialization function
- Pull test data
- Validate data
- Return passed/failed

```
    while(testCounter<15)
    {
        ++testCounter;
        AccelInit();
        getAccel();
        if(Accelleration.x>=-10 &&
            Accelleration.x<=10 &&
                Accelleration.y>=-10 &&
                    Accelleration.y<=10)
        {
            passed=1;
            break;
        }
    }

    if(passed)
        return 1;
    else
        return 0;
}
```

Sensor Boot Sequence

- Error handling
- Call initialization function
- Pull test data
- Validate data
- Return passed/failed

```
while(testCounter<15)
{
    ++testCounter;
    AccelInit();
    getAccel();
    if(Accelleration.x>=-10 &&
        Accelleration.x<=10 &&
            Accelleration.y>=-10 &&
                Accelleration.y<=10)
    {
        passed=1;
        break;
    }
}

if(passed)
    return 1;
else
    return 0;
}
```

Sensor Boot Sequence

- Error handling
- Call initialization function
- Pull test data
- Validate data
- Return passed/failed

```
    while(testCounter<15)
    {
        ++testCounter;
        AccelInit();
        getAccel();
        if(Accelleration.x>=-10 &&
            Accelleration.x<=10 &&
                Accelleration.y>=-10 &&
                    Accelleration.y<=10)
        {
            passed=1;
            break;
        }
    }

    if(passed)
        return 1;
    else
        return 0;
}
```

Sensor Boot Sequence

- Error handling
- Call initialization function
- Pull test data
- Validate data
- Return passed/failed

```
  while(testCounter<15)
  {
      ++testCounter;
      AccelInit();
      getAccel();
      if(Accelleration.x>=-10 &&
         Accelleration.x<=10 &&
            Accelleration.y>=-10 &&
               Accelleration.y<=10)
      {
         passed=1;
         break;
      }
  }

  if(passed)
      return 1;
  else
      return 0;
}
```

Sensor Boot Sequence

- Error handling
- Call initialization function
- Pull test data
- Validate data
- Return passed/failed

```
  while(testCounter<15)
  {
      ++testCounter;
      AccelInit();
      getAccel();
      if(Accelleration.x>=-10 &&
          Accelleration.x<=10 &&
              Accelleration.y>=-10 &&
                  Accelleration.y<=10)
      {
          passed=1;
          break;
      }
  }

  if(passed)
      return 1;
  else
      return 0;
}
```

User Interface/Configuration

Must provide instant indication of status for all sensors

●	**Power**
●	**GPS Signal**
●	**SD Card**
●	**OBDII Data**
●	**Camera**
●	**Expansion Port**

Display Status:

- ● Normal Operation
- ✳ Low Power
- ● Not Connected
- ● No Signal
- ✳ Media Not Present
- ✳ Media Full
- ✳ Media Unwritable
- ✳ Communications Error

- **Configuration will be done with text file located on the SD card**
- **If config file is not present at startup one will be created with default values**
- **Configuration values such as output format and logging frequencies will be editable via text editor.**

User Interface

- Software currently flashes sensor's LED upon activity for debugging purposes
- Final UI not yet completed

```
int getYawRate()
{
    int i;
    int DOUT[17];
    int decimalSum;

    DOUT[0]=0;

    WrPortI(PEB5R, NULL, 0xFF);
    BitWrPortI(PBDR, PBDRShadow,  0, 7);    //CS low
    BitWrPortI(PCDR, PCDRShadow,  1, 4);    //WRITE


    decimalSum += DOUT[5]*1024;
    decimalSum += DOUT[4]*2048;

    WrPortI(PEB5R, NULL, 0x00);

    return decimalSum;
}
/* END: YAW RATE GYRO FUNCTIONS */
```

User Interface

- Indicate sensor activity
- Mark end of boot sequence
- Notify user of any sensor initialization failures

```
printf("\nMain Function Starting");
printf("\nChecking Devices...");

OBDGTG = prepOBD();
yawGTG = prepYaw();
accelGTG = prepAccel();
GPSGTG = prepGPS();


flashLED1();
flashLED2();
flashLED3();
flashLED4();
flashLED5();


if(accelGTG==0)
    WrPortI(PEB3R, NULL, 0xFF);
if(OBDGTG==0)
    WrPortI(PEB4R, NULL, 0xFF);
if(yawGTG==0)
    WrPortI(PEB5R, NULL, 0xFF);
if(camGTG==0)
    WrPortI(PEB6R, NULL, 0xFF);
if(GPSGTG==0)
    WrPortI(PEB7R, NULL, 0xFF);
```

User Interface

- Indicate sensor activity
- Mark end of boot sequence
- Notify user of any sensor initialization failures

```
printf("\nMain Function Starting");
printf("\nChecking Devices...");

OBDGTG = prepOBD();
yawGTG = prepYaw();
accelGTG = prepAccel();
GPSGTG = prepGPS();


flashLED1();
flashLED2();
flashLED3();
flashLED4();
flashLED5();


if(accelGTG==0)
    WrPortI(PEB3R, NULL, 0xFF);
if(OBDGTG==0)
    WrPortI(PEB4R, NULL, 0xFF);
if(yawGTG==0)
    WrPortI(PEB5R, NULL, 0xFF);
if(camGTG==0)
    WrPortI(PEB6R, NULL, 0xFF);
if(GPSGTG==0)
    WrPortI(PEB7R, NULL, 0xFF);
```

User Interface

- Indicate sensor activity
- Mark end of boot sequence
- Notify user of any sensor initialization failures

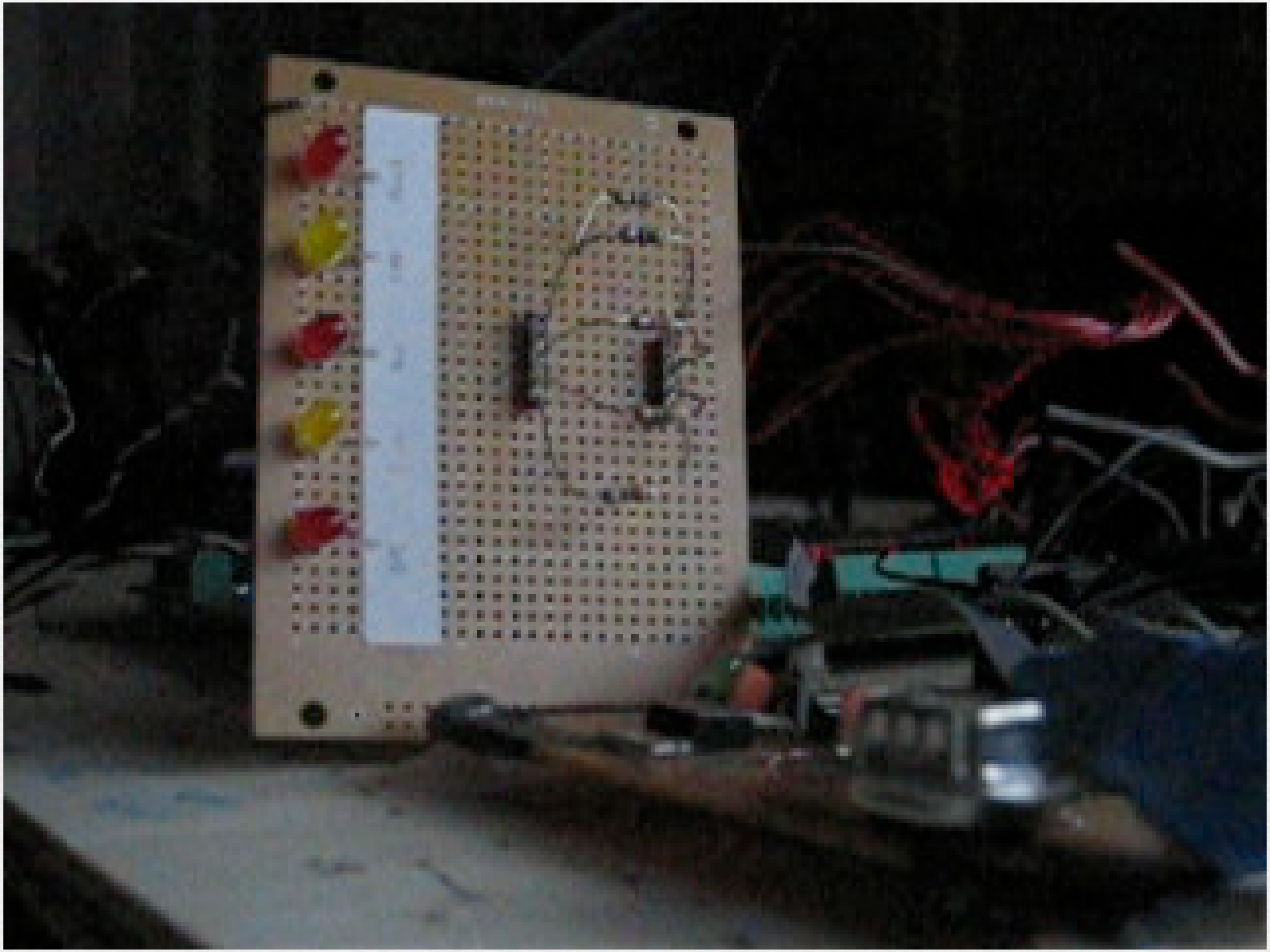
```
printf("\nMain Function Starting");
printf("\nChecking Devices...");

OBDGTG = prepOBD();
yawGTG = prepYaw();
accelGTG = prepAccel();
GPSGTG = prepGPS();

flashLED1();
flashLED2();
flashLED3();
flashLED4();
flashLED5();

if(accelGTG==0)
    WrPortI(PEB3R, NULL, 0xFF);
if(OBDGTG==0)
    WrPortI(PEB4R, NULL, 0xFF);
if(yawGTG==0)
    WrPortI(PEB5R, NULL, 0xFF);
if(camGTG==0)
    WrPortI(PEB6R, NULL, 0xFF);
if(GPSGTG==0)
    WrPortI(PEB7R, NULL, 0xFF);
```

User Interface

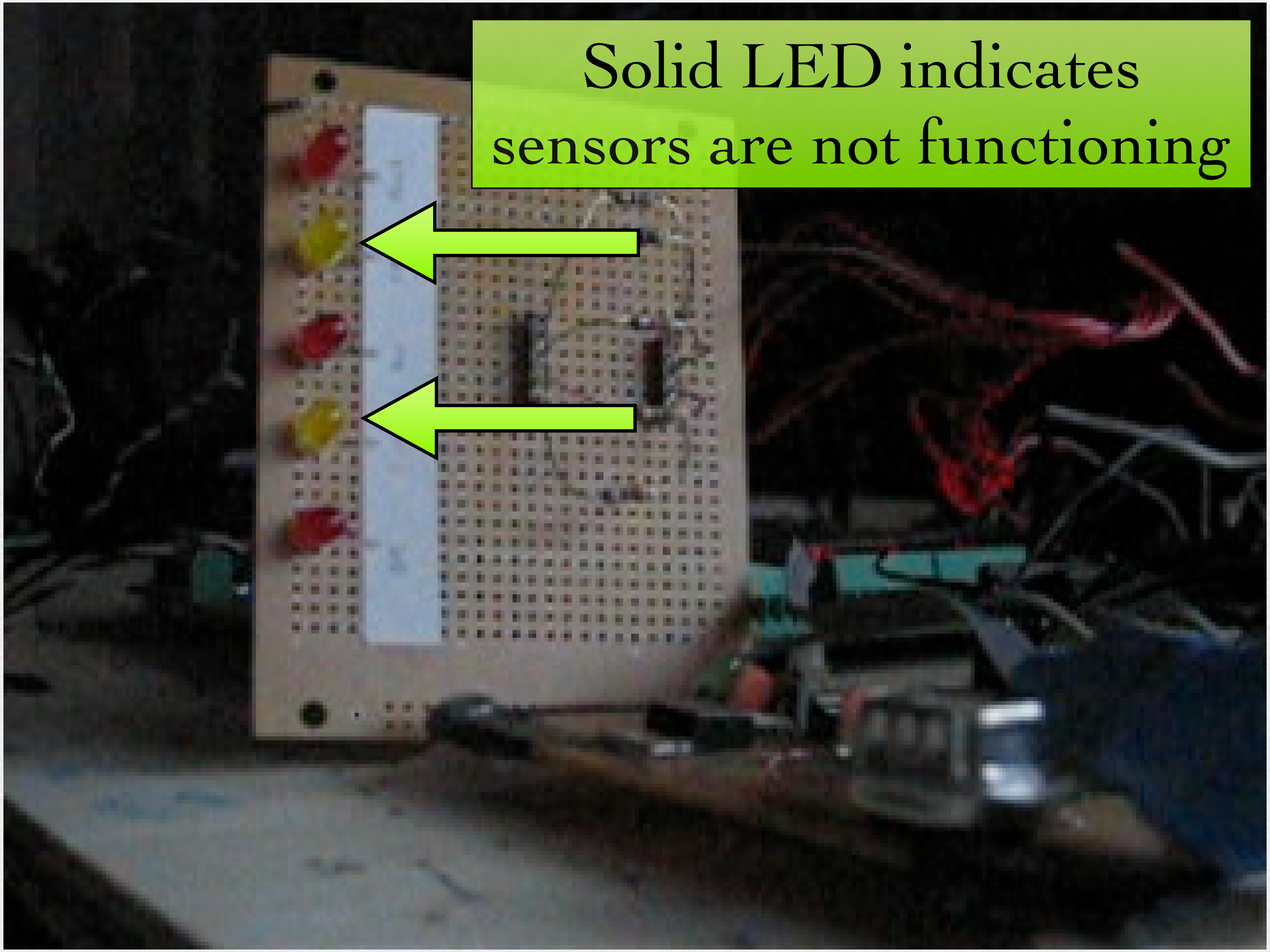
- Indicate sensor activity
- Mark end of boot sequence
- Notify user of any sensor initialization failures

```
printf("\nMain Function Starting");
printf("\nChecking Devices...");

OBDGTG = prepOBD();
yawGTG = prepYaw();
accelGTG = prepAccel();
GPSGTG = prepGPS();


flashLED1();
flashLED2();
flashLED3();
flashLED4();
flashLED5();


if(accelGTG==0)
    WrPortI(PEB3R, NULL, 0xFF);
if(OBDGTG==0)
    WrPortI(PEB4R, NULL, 0xFF);
if(yawGTG==0)
    WrPortI(PEB5R, NULL, 0xFF);
if(camGTG==0)
    WrPortI(PEB6R, NULL, 0xFF);
if(GPSGTG==0)
    WrPortI(PEB7R, NULL, 0xFF);
```

User Interface Video



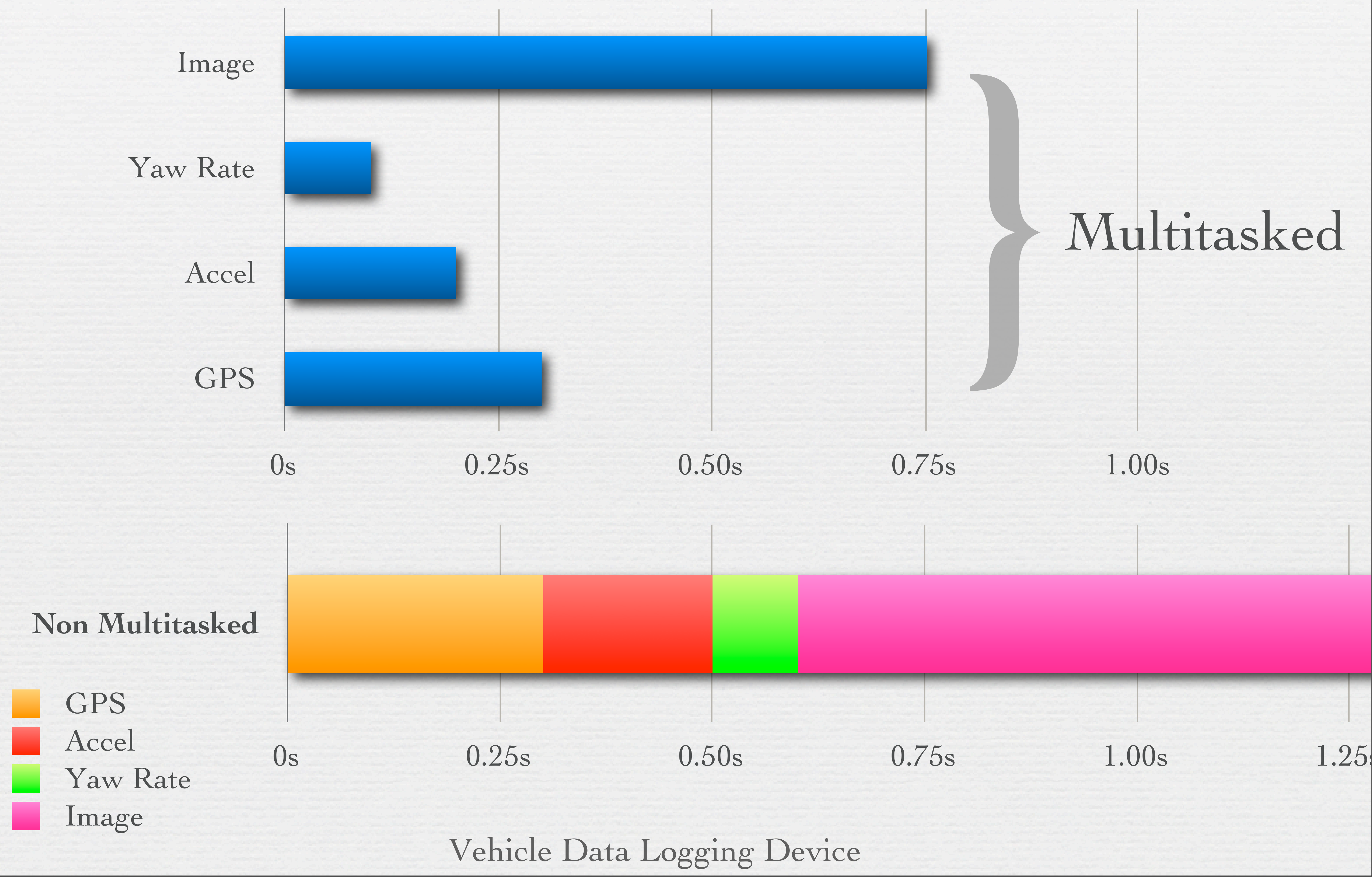
User Interface Video



Sensor Polling

The need for multitasking.

Multitasking Timing Illustration



Polling Sensors

Multitasking

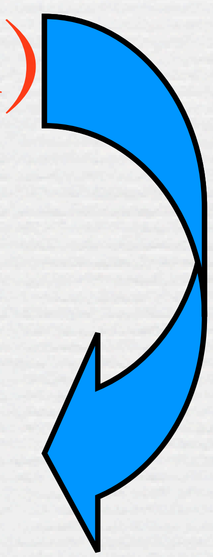
```
while(1){
  costate { … }
  costate { … }
  costate { … }
}
```

- costate descriptor
- Resides within infinite loop
- costate flow proceeds in three ways

Polling Sensors

Multitasking

```
costate{
  statement 1
  statement 2
  waitfor(condition)
  statement 3
  statement 4
}
```

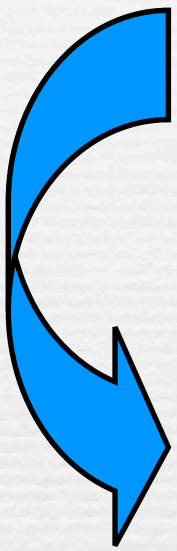


Initial entry into costate

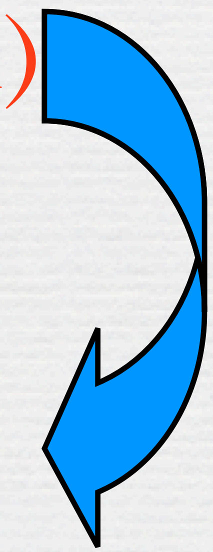
- Statement 1 & 2 executed
- waitfor evaluated to zero (0)
- Flow jumps out of the costate

Polling Sensors

Multitasking



```
costate{
    statement 1
    statement 2
    waitfor(condition)
    statement 3
    statement 4
}
```



Subsequent entry into costate

- **Flow jumps directly to waitfor**
- **waitfor evaluated to zero (0)**
- **Flow jumps out of the costate**

Polling Sensors

Multitasking

```
costate{
    statement 1
    statement 2
    waitfor(condition)
    statement 3
    statement 4
}
```

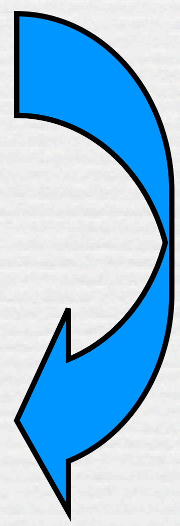
Subsequent entry into costate

- **Flow jumps directly to waitfor**
- **waitfor evaluated to one (1)**
- **Statement 3 & 4 executed**
- **System reference to costate reset**

Polling Sensors

Multitasking

```
costate{
   statement 1
   statement 2
   waitfor(condition)
   statement 3
   statement 4
}
```



<u>After costate reference is reset</u>

- **Next cycle will be executed as an initial entry**
- **Cycle starts anew**
- **Statement 1 & 2 executed**
- **waitfor evaluated to zero (0)**

Polling Sensors

Multitasking

- Allows Rabbit to devote time to other tasks while…
  - Waiting for ELM327 Ready to Receive Flag
  - Waiting for ELM327 to fill buffer with data

```
cofunc int getThrot()
{
    char readSentence[25];
    char OBDresponse[5];
    int carThrot;
    int x;

    carThrot=1;

    serErdFlush();
    serEwrFlush();
    waitfor(0==BitRdPortI(PGDR, 4));
    serEputs("01 11\r");
    waitfor(20==serErdUsed());
    serEread(readSentence, 20, 10);
```

Polling Sensors

Multitasking

- Allows Rabbit to devote time to other tasks while…
  - Waiting for ELM327 Ready to Receive Flag
  - Waiting for ELM327 to fill buffer with data

```
cofunc int getThrot()
{
   char readSentence[25];
   char OBDresponse[5];
   int carThrot;
   int x;

   carThrot=1;

   serErdFlush();
   serEwrFlush();
   waitfor(0==BitRdPortI(PGDR, 4));
   serEputs("01 11\r");
   waitfor(20==serErdUsed());
   serEread(readSentence, 20, 10);
```

Polling Sensors

Multitasking

- Allows Rabbit to devote time to other tasks while…
    - Waiting for ELM327 Ready to Receive Flag
    - Waiting for ELM327 to fill buffer with data

```
cofunc int getThrot()
{
   char readSentence[25];
   char OBDresponse[5];
   int carThrot;
   int x;

   carThrot=1;

   serErdFlush();
   serEwrFlush();
   waitfor(0==BitRdPortI(PGDR, 4));
   serEputs("01 11\r");
   waitfor(20==serErdUsed());
   serEread(readSentence, 20, 10);
```

Storing Data to SD Card

- New text file is created on system boot
    - system boots each time ignition is cycled
- Each cycle's data is appended to this text file

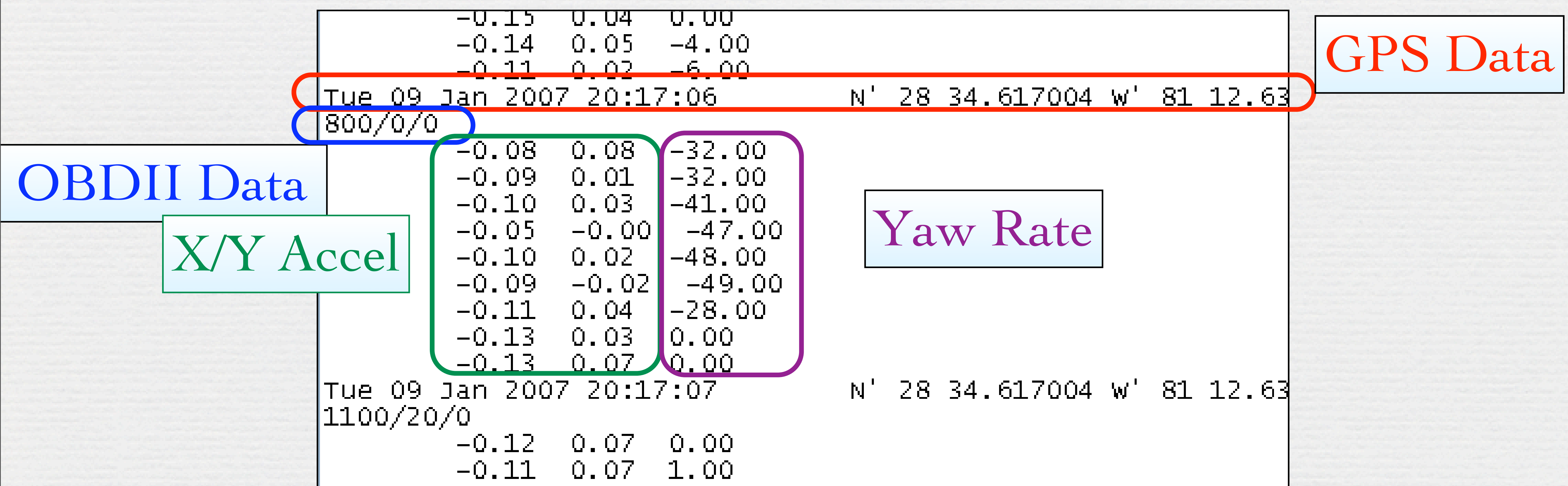


Image Capture

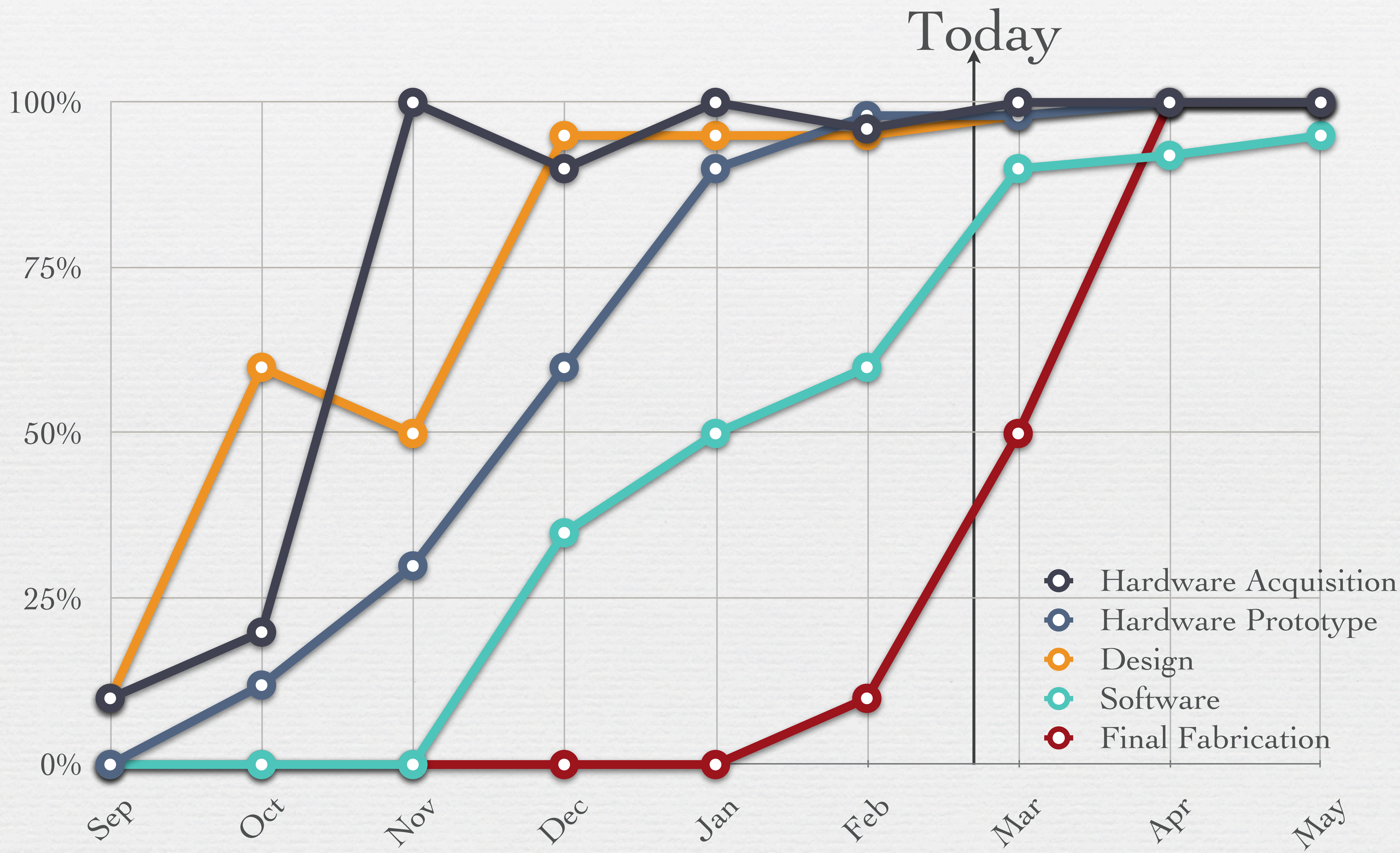
- Stored to a separate folder
  - **folder named by trip start-time**
  - **images named by time**
- Image quality will be improved
  - **currently 80x60**
  - **will be min. 160x120**

Software to Do

- Improve Image Capture
- Complete Final User Interface
- Implement System Configurability
    - **Allow user to edit text file on SD to change system settings**
- Incorporate Intelligent Power Supply
- Finalize Code for Stand Alone Mode

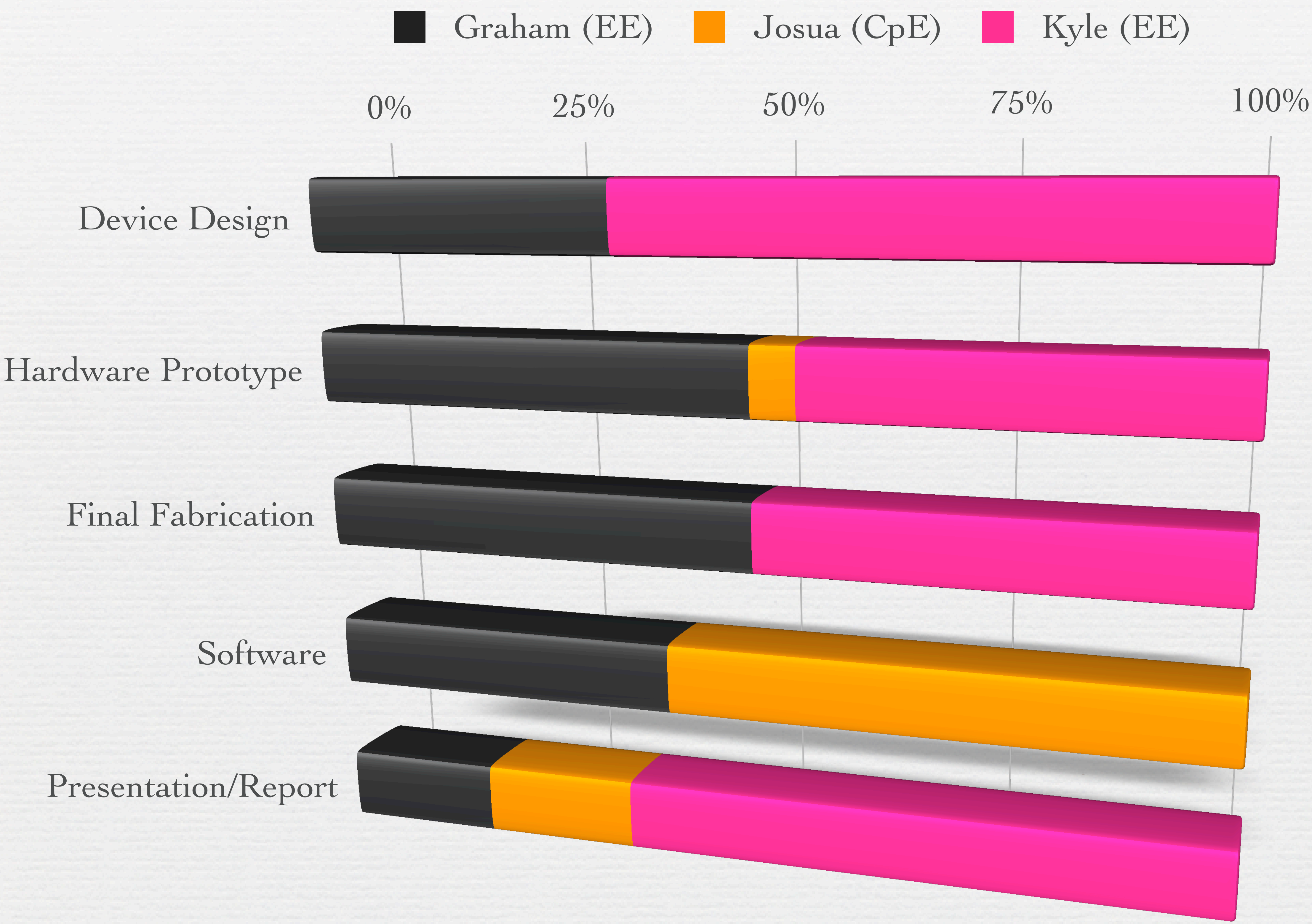
Progress & Budget

Project Progress



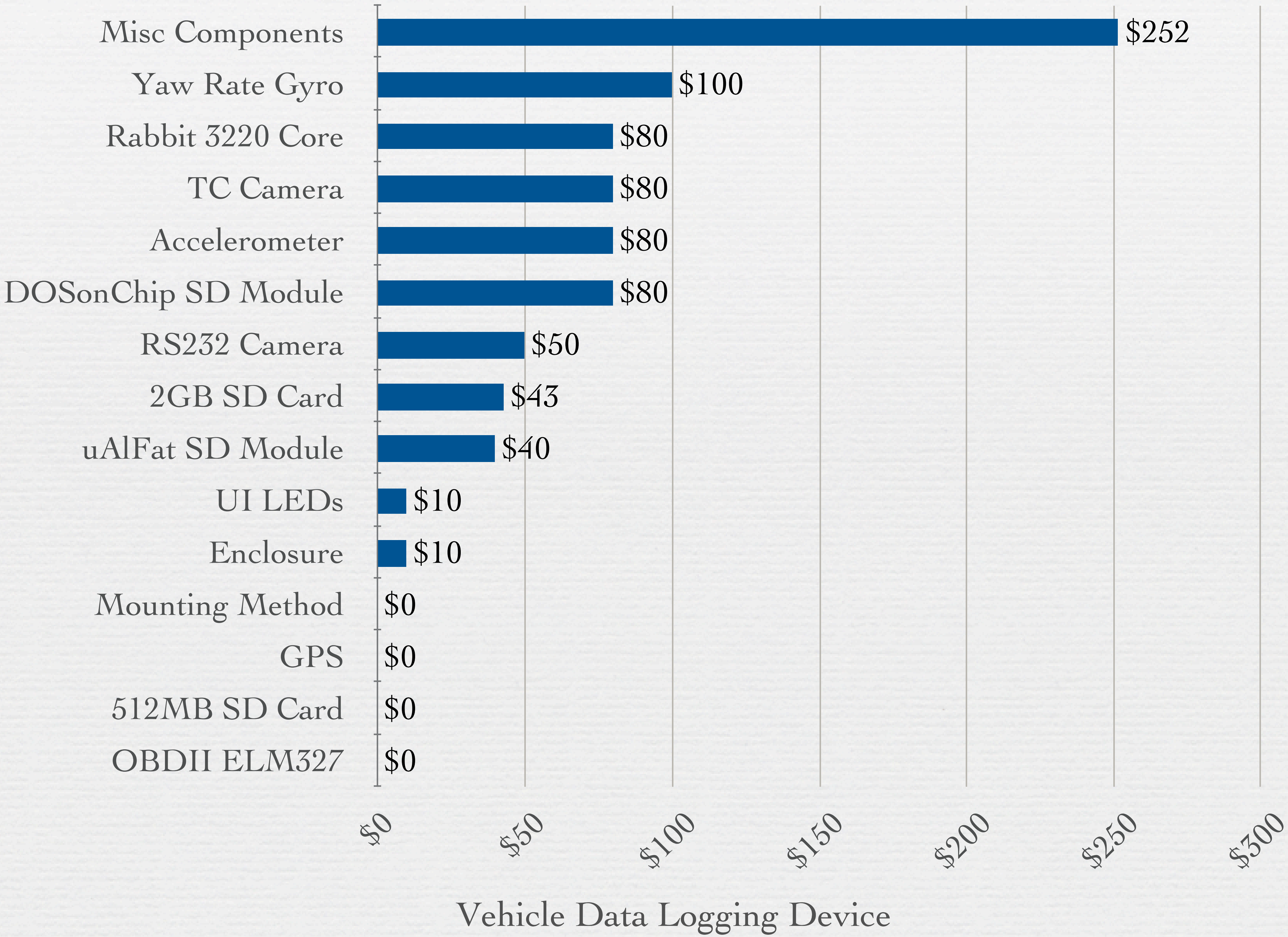
Vehicle Data Logging Device

Work Breakdown



Vehicle Data Logging Device

Budget


Misc Components
Yaw Rate Gyro
Rabbit 3220 Core
TC Camera
Accelerometer
DOSonChip SD Module
RS232 Camera
2GB SD Card
uAlFat SD Module
UI LEDs
Enclosure
Mounting Method
GPS
512MB SD Card
OBDII ELM327
$252
$100
$80
$80
$80
$80
$50
$43
$40
$10
$10
$0
$0
$0
$0
$0
$50
$100
$150
$200
$250
$300
Vehicle Data Logging Device

Miscellaneous Costs

Capacitors, power regulators, wire, perf board, tools, solder, SOIC adapters, DIP adapters, diodes, transistors, DB9 connectors, markers, PCB etching materials, pre-made breakout boards, RS232 level shifters, breadboards, test leads, and more...

Budget Specifics

Part	Actual Cost	Our Cost	# Acquired	Total Spent
GPS Device	$80.00	$0.00	2	$0.00
CMOS Image Sensor	$40.00	$40.00	2	$80.00
RS232 Camera	$50.00	$50.00	1	$50.00
Accelerometer	$40.00	$40.00	2	$80.00
Yaw Rate Gyro	$50.00	$50.00	2	$100.00
ELM327	$35.00	$0.00	2	$0.00
DOSonChip	$40.00	$40.00	2	$80.00
uALFAT	$40.00	$40.00	2	$80.00
UI LEDs (4)	$10.00	$10.00	0	$0.00
2GB SD Card	$43.00	$43.00	1	$43.00
512MB SD Card	$20.00	$0.00	1	$0.00
Rabbit 3220 Core	$80.00	$80.00	1	$80.00
Misc Components	??	$251.60	1	$251.60
Enclosure	$10.00	$10.00	0	$0.00
Mounting Method	$5.00	$5.00	0	$0.00
Parts Totals	$455.00	$659.60	19	$844.60

Device Reproduction Costs

Required Parts	Actual Cost
Rabbit 3220 Core	$80.00
Accelerometer	$15.00
ELM327	$35.00
GPS Device	$80.00
512MB SD Card	$20.00
uALfat	$40.00
Misc Components	$30.00
Required Parts Total	$300.00
Camera	$40.00
Yaw Rate Gyro	$50.00
Parts Totals	$390.00

Without optional components, the total cost of fabrication is easily within the budget constraints.

With optional components we must have a large production run in order to meet the specified system cost.

We suggest in this instance the Rabbit be swapped for a PIC thereby saving $80, about the cost of the additional components.

Number of Boards	2	4	6	8	10	100
PCB Fabrication	$222.00	$256.00	$282.00	$308.00	$334.00	$1076.00
Cost/Board	$111.00	$64.00	$47.00	$38.50	$33.40	$10.76
Totals	$501.00	$454.00	$437.00	$428.50	$423.40	$400.76
w/o opt components	$411.00	$364.00	$347.00	$338.50	$333.40	$310.76

Successes

- Hardware Acquired
- Prototype Built and Tested
- Device is logging data from all sensors
- Output Verified as Accurate
- Software 80% Complete
- Final Hardware Schematic Finished
- Beginning PCB Design

Difficulties

- LGA 16 package (yaw rate gyro) was difficult to work with
- SPI driver had to be rewritten
- 24 Pin ZIF (first camera) also difficult to work with
- Hardware discontinued mid-development (first camera)
- Reworking storage device due to insufficient write speed (DOSonCHIP)

Future Progress

- Optimize Software
- Complete Final User Interface
- Incorporate Intelligent Power Supply
- Improve Image Capture
- Implement System Configurability
- Finalize Code for Stand-Alone Mode
- Reliability Test Device
- Design & Order PCB
- Enclosure
- Final Testing

Questions?