ISOBlue

Alex Layton

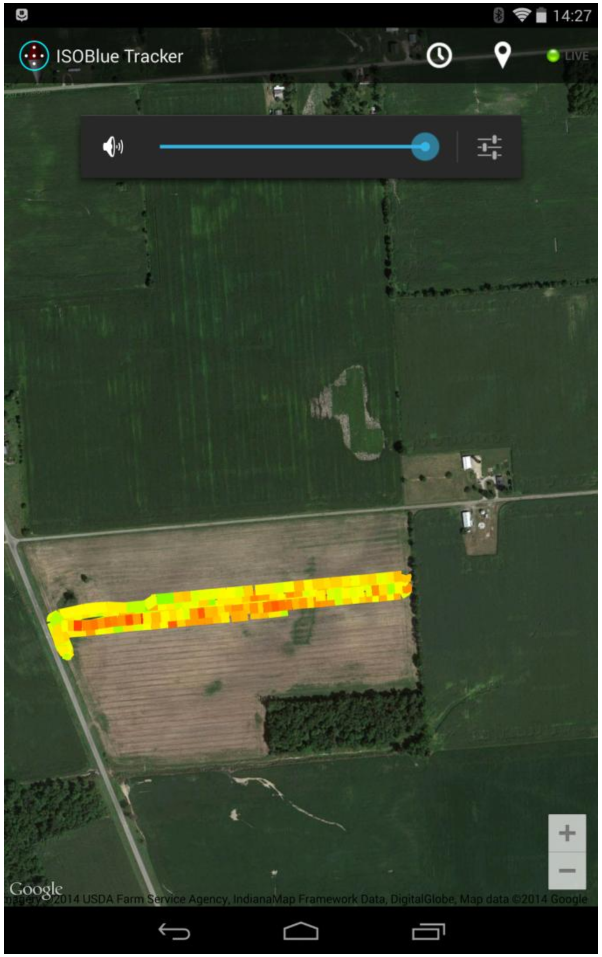
Pat Sabpisal

Purdue University

25 April 2014

Yield Monitor App

- Written by Pat Sabpisal
- Receives GNSS and “Grain Flow” ISOBUS messages using *libISOBlue* Android library
- Uses the received ISOBUS data to generate a yield map and shows it on a Google Map
- Uses experimentally determined conversion to convert gain flow measurement to bushels per second
- Updates the yield map in real-time as ISOBUS data is received


14:27
ISOBlue Tracker
LIVE
Google
Imagery ©2014 USDA Farm Service Agency, IndianaMap Framework Data, DigitalGlobe, Map data ©2014 Google

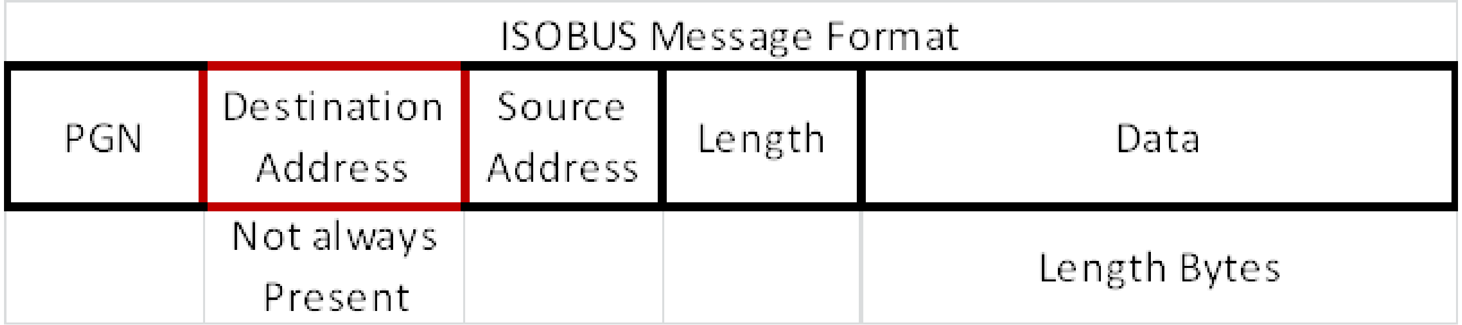
ISOBlue Demo

- Two ISOBlues are connected with ISOBUS
  - One is the real ISOBlue
  - One is the "combine"
- Via terminal, the "combine" is made to resend recorded ISOBUS data
- An Android tablet running the Yield Monitor App will connect to ISOBlue and generate a yield map
- The app will be restarted midway through the data set
  - The app will receive the beginning data from ISOBlue's buffer
  - It will receive the remaining data in "real-time"

A recoding of the app during a run of the demo is here https://www.youtube.com/watch?v=5FSMPHDJ5RE

https://www.youtube.com/watch?v=qLbVUeMaR_8

ISOBUS Overview



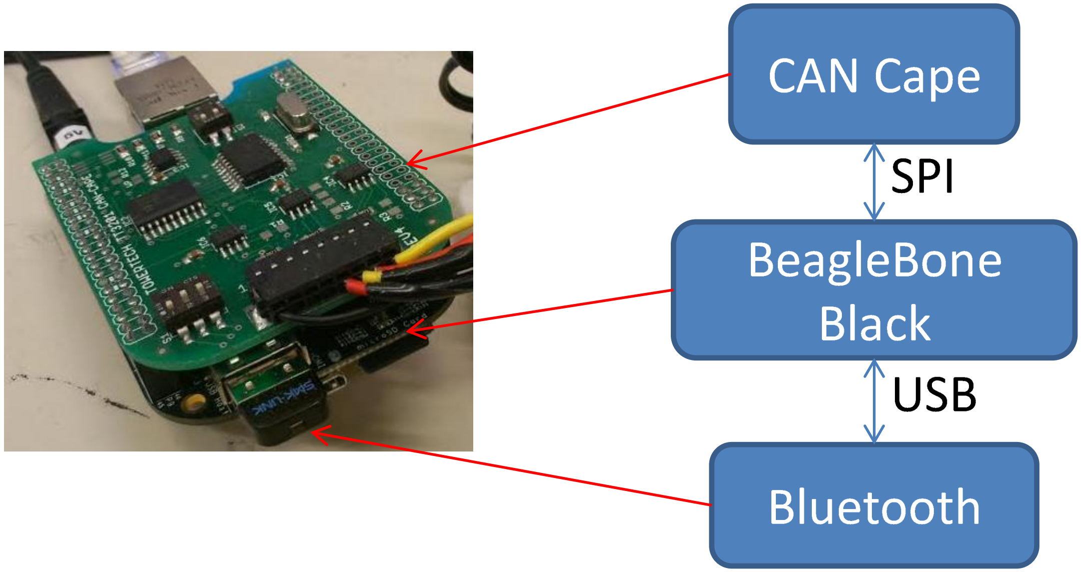
ISOBUS Message Format

PGN	Destination Address	Source Address	Length	Data
	Not always Present			Length Bytes

- Communication protocol used in the agricultural industry
- The ISOBUS network is composed of two separate CAN busses
  - Tractor (Engine) Bus
  - Implement Bus
- The ISOBUS bitrate is 250 kbps (per bus)
- ISOBUS data is sent using messages with the above format
- The data of a message is identified by its PGN (Parameter Group Number)
  - To interpret a message's data, one must have access to the specification of its corresponding Parameter Group

ISOBlue Device

- Platform is a BeagleBone Black
  - Runs Angstrom, a kind of Linux
- Connects to the ISOBUS network with a CAN cape which has multiple CAN interfaces
  - Cape is compatible with SocketCAN
- Talks to Android over Bluetooth to forward ISOBUS messages
  - Uses USB Bluetooth Dongle



CAN Cape

SPI

BeagleBone Black

USB

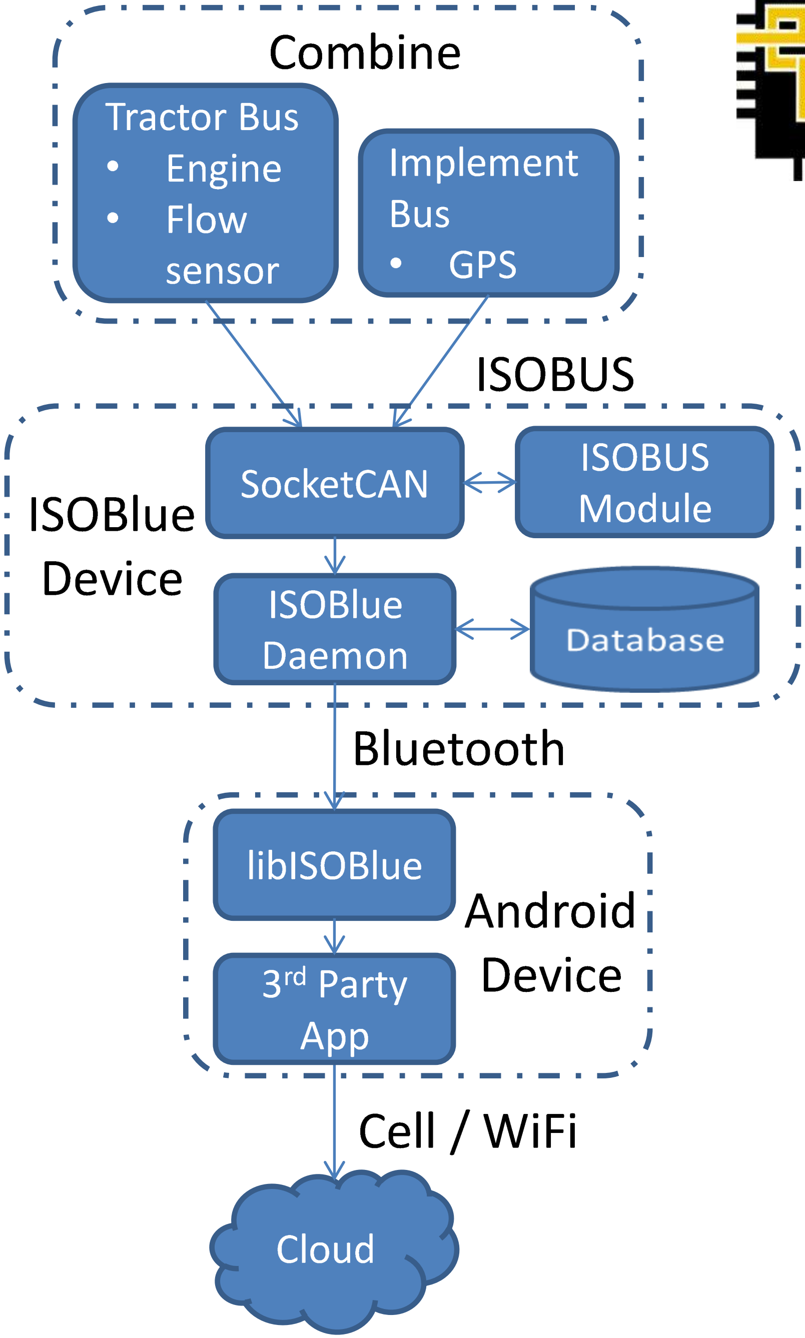
Bluetooth

System Overview

Main Components

- ISOBlue device
- ISOBlue Daemon software
- *libISOBlue* Android library

ISOBlue Data Flow



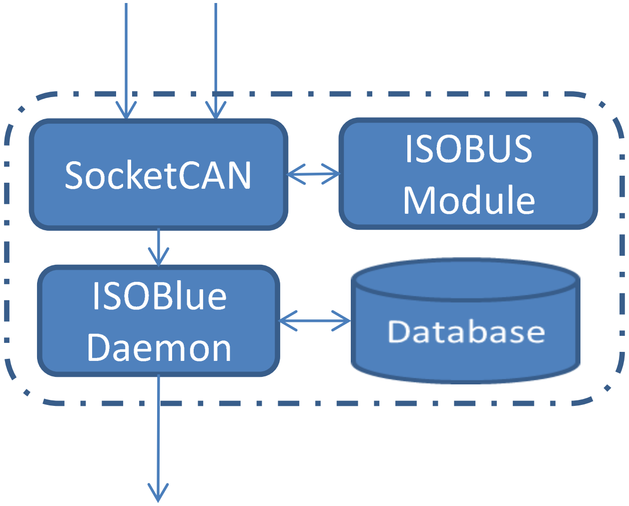
ISOBlue Daemon Software

Software written to work as a "server" for the Android library

- Runs on ISOBlue, waiting for an Android device to connect using *libISOBlue*
- Monitors ISOBUS network with *SocketCAN* and ISOBUS kernel module
  - SocketCAN is the de facto standard CAN driver for Linux
  - The ISOBUS kernel module is an addition to this driver, which runs as part of Linux rather than as an application
- Stores received messages in a database
  - Database used is LevelDB from Google
  - Its purpose it to efficiently store and retrieve messages on the SD card
- Forwards ISOBUS messages from SocketCAN and/or the database to *libISOBlue*

Data Flow

ISOBUS from combine



Bluetooth to *libISOBlue*

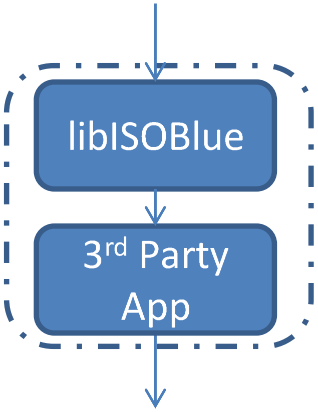
libISOBlue Android Library

Custom written in Java, runs on the Android device

- Receives ISOBUS messages forwarded by ISOBlue
  - Talks to ISOBlue over Bluetooth
  - Tries to automatically reconnect when connection is lost
- Presents a simple socket like interface to the app using it
  - At a basic level a socket is an object which when you read it you are returned a piece of data someone else sent to you since the last time you read the socket
  - Sockets can be used to read data off either bus of the ISOBUS network
  - Buffered sockets can be used to read data from before the Android device connected to ISOBlue
  - Sockets give PGN, SA, DA, data bytes, and timestamp for each message
- The library can be used to set filters on which ISOBUS messages ISOBlue receives
  - Allows choosing which PGNs to receive and on which bus

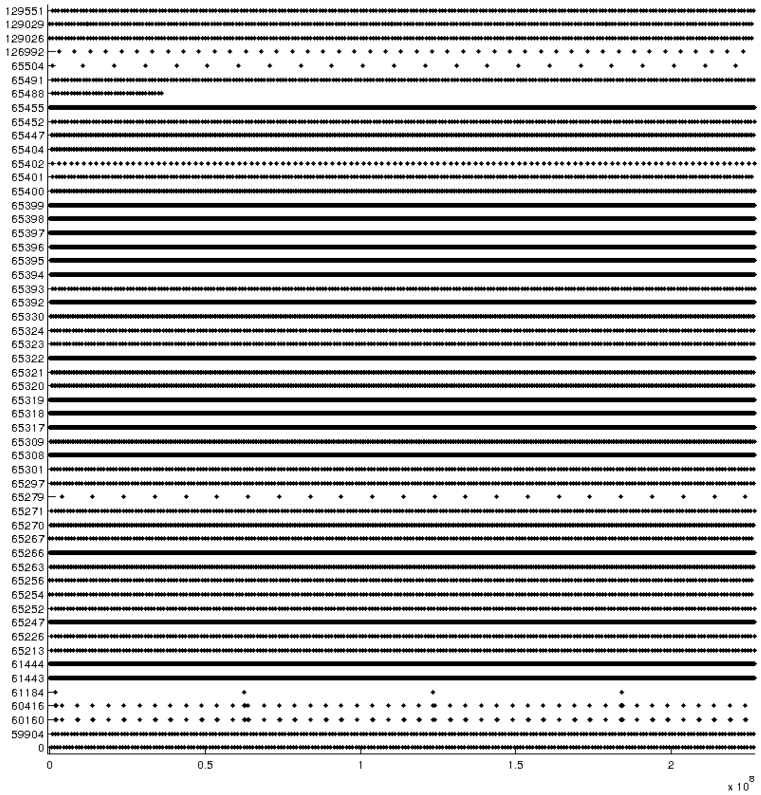
Data Flow

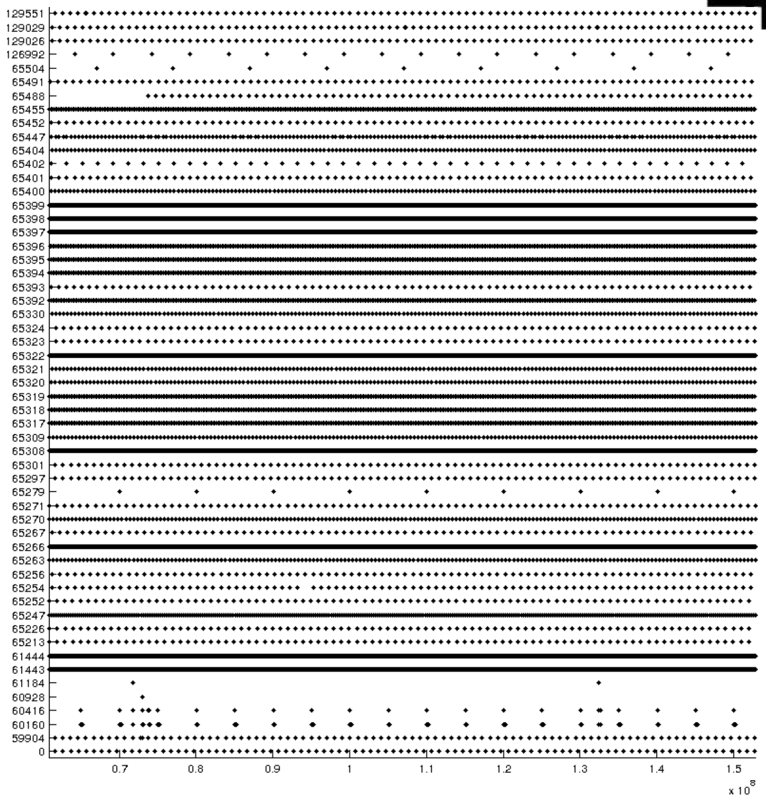
Bluetooth from *ISOBlue*



Cell/WiFi to Cloud

Finding Grain Flow Message



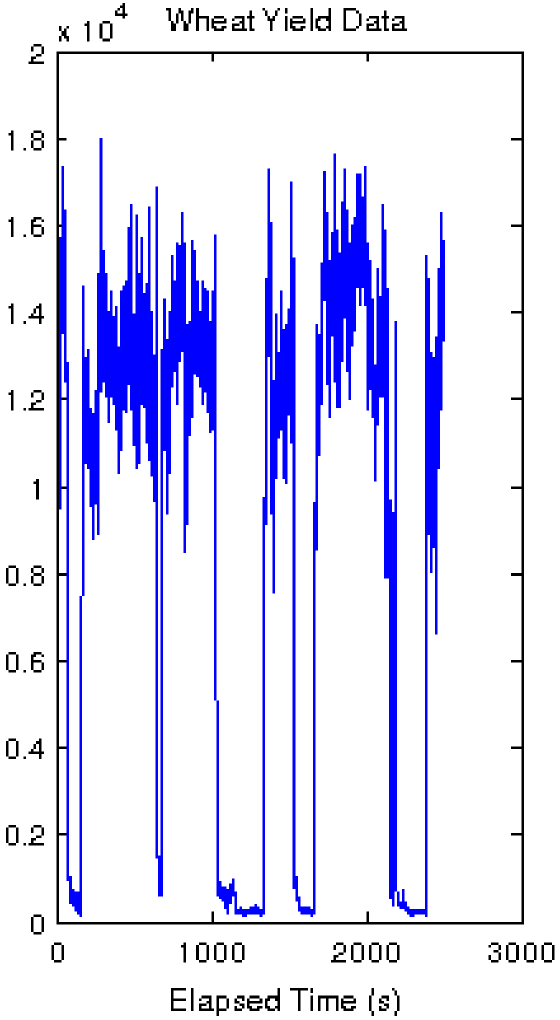


- Unplugged and plugged back in the grain flow sensor while logging ISOBUS messages
- Found one PGN stopped/started in correspondence

Examining Grain Flow Message

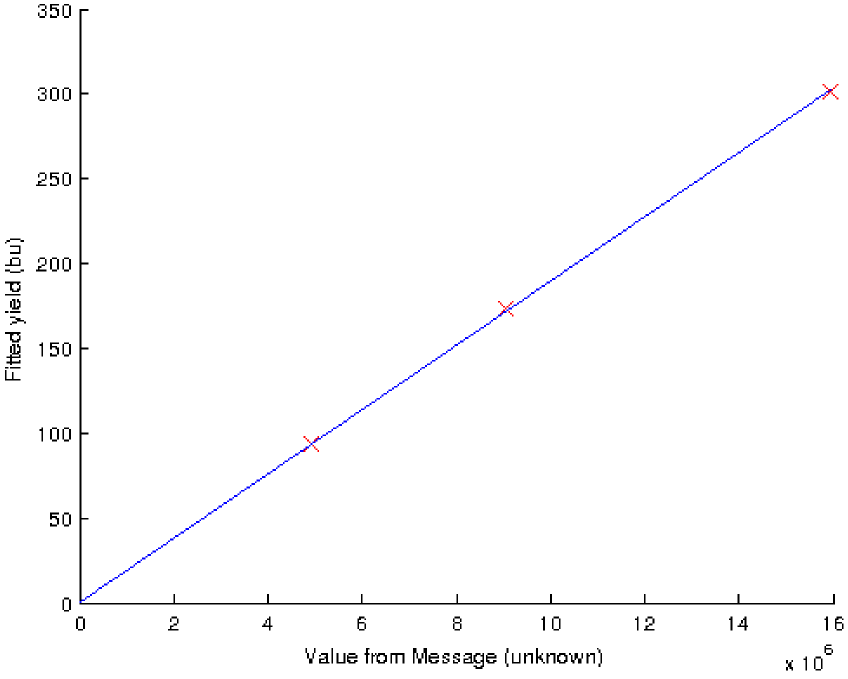
- Looked at the data bytes of messages with the “grain flow” PGN
- Noticed two portions of the bytes which were changing
  - First two bytes
  - Last two bytes
    - Went to zero when the combine was not harvesting
- Concluded last two data bytes of the message were the flow measurement

Message Data Bytes
DCAD0000FFFFFBDF
F9000000FFFFB36D
06010000FFFF8871
B6000000FFFF3079
C8000000FFFF047D
9B000000FFFFD780
BA000000FFFFAC84
B1000000FFFF8088
A4000000FFFF538C
13010000FFFF2890
C3000000FFFFFC93
CC000000FFFFCF97
97000000FFFFA49B
C3000000FFFF779F
F4000000FFFF4BA3
E2000000FFFF1FA7
37010000FFFFF3AA



Determining Yield Conversion

- The conversion of the flow sensor measurement to bushels needed to be determined
- The yield messages for a section of a field were logged along with what the monitor reported as the total bushels for that area
- Using the monitor as truth, least squares was used to determine the appropriate scaling for the flow measurements
- The fit turned out to be quite good
  - The errors of the fitted points were all under 1%

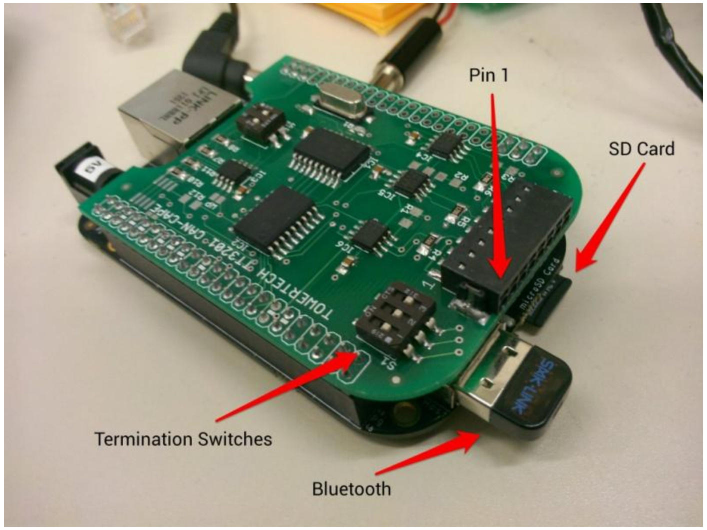


Building an ISOBlue

ISOBlue can be assembled without tools.

Parts:

- BeagleBone Black
- USB Bluetooth Dongle
- CAN Cape
- SD Card



Steps:

- Load Angstrom Linux onto SD card
- Insert SD card and USB Bluetooth dongle, and attach CAN cape
- Boot Linux
- Install ISOBlue software
  - Clone GitHub repository
  - Follow contained instructions for compiling and installing (found in README file)

Detailed tutorial available at:
https://github.com/ISOBlue/isoblue-software/tree/master/tutorial

Project Contributions

- Found appropriate parts and assembled first ISOBlue prototype
- Wrote ISOBUS kernel module for SocketCAN
- Wrote ISOBlue Daemon software
- Wrote Angstrom Linux configuration files for making ISOBlue work at startup
- Wrote *libISOBlue* Android library
- Determined which ISOBUS messages contain grain flow measurements
- Determined how to interpret grain flow measurements obtained from ISOBUS messages

To Do

- ISOBlue cape
  - Design the cape
  - Create prototype
  - Write software taking advantage of the new cape
- Grain moisture message
  - Find which PGN corresponds to it
  - Determine how to interpret its data
- ISOBUS kernel module for use with SocketCAN
  - Add support for ISOBUS transport protocols

Proposed ISOBlue Cape

Total estimated cost for one cape ~$330

Proposed ISOBlue Cape

Total estimated cost for one cape ~$330

- Cell Modem (MTSMC-H5-IP) ~$200

Image from digikey.com

Proposed ISOBlue Cape

Total estimated cost for one cape ~$330

- Cell Modem (MTSMC-H5-IP) ~$200
- GPS Receiver (NV08C-CSM) ~$50
  - Needs external antenna

Image from newark.com

Proposed ISOBlue Cape

Total estimated cost for one cape ~$330

- Cell Modem (MTSMC-H5-IP) ~$200
- GPS Receiver (NV08C-CSM) ~$50
  - Needs external antenna
- Provide Power from ISOBUS (PYB10-Q24-S5) ~$24
  - Needs circuitry to turn ISOBlue on/off based on when the ISOBUS is active

Image from laddinc.com

Proposed ISOBlue Cape

Total estimated cost for one cape ~$330

- Cell Modem (MTSMC-H5-IP) ~$200
- GPS Receiver (NV08C-CSM) ~$50
  - Needs external antenna
- Provide Power from ISOBUS (PYB10-Q24-S5) ~$24
  - Needs circuitry to turn ISOBlue on/off based on when the ISOBUS is active
- CAN Ports ~$12
  - 2 x Transceiver (SN65HVD232) ~$3
  - 2 x Controller (MCP2515) ~$3
  - Could keep using current CAN cape instead

Proposed ISOBlue Cape

Total estimated cost for one cape ~$330

- Cell Modem (MTSMC-H5-IP) ~$200
- GPS Receiver (NV08C-CSM) ~$50
  - Needs external antenna
- Provide Power from ISOBUS (PYB10-Q24-S5) ~$24
  - Needs circuitry to turn ISOBlue on/off based on when the ISOBUS is active
- CAN Ports ~$12
  - 2 x Transceiver (SN65HVD232) ~$3
  - 2 x Controller (MCP2515) ~$3
  - Could keep using current CAN cape instead
- PCB ~$20
  - Hard to know this price well until the board is designed

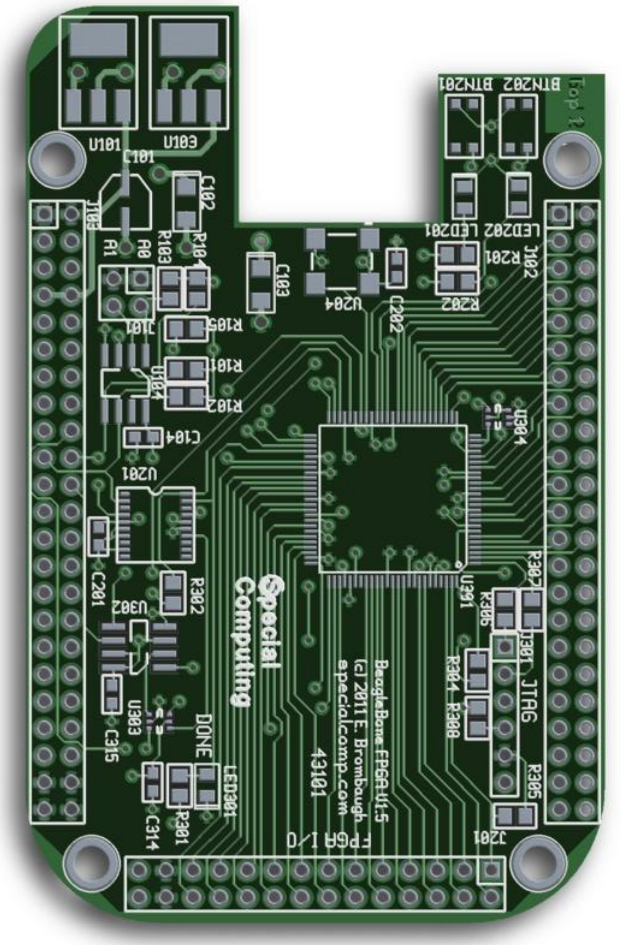


Image from specialcomp.com

Proposed ISOBlue Cape

Total estimated cost for one cape ~$330

- Cell Modem (MTSMC-H5-IP) ~$200
- GPS Receiver (NV08C-CSM) ~$50
  - Needs external antenna
- Provide Power from ISOBUS (PYB10-Q24-S5) ~$24
  - Needs circuitry to turn ISOBlue on/off based on when the ISOBUS is active
- CAN Ports ~$12
  - 2 x Transceiver (SN65HVD232) ~$3
  - 2 x Controller (MCP2515) ~$3
  - Could keep using current CAN cape instead
- PCB ~$20
  - Hard to know this price well until the board is designed
- Materials etc. ~$20

Image from tangentindinc.com

Proposed ISOBlue Cape

Total estimated cost for one cape ~$330

- Cell Modem (MTSMC-H5-IP) ~$200
- GPS Receiver (NV08C-CSM) ~$50
  - Needs external antenna
- Provide Power from ISOBUS (PYB10-Q24-S5) ~$24
  - Needs circuitry to turn ISOBlue on/off based on when the ISOBUS is active
- CAN Ports ~$12
  - 2 x Transceiver (SN65HVD232) ~$3
  - 2 x Controller (MCP2515) ~$3
  - Could keep using current CAN cape instead
- PCB ~$20
  - Hard to know this price well until the board is designed
- Materials etc. ~$20

Cape could be partially populated with components to reduce cost when only a subset of the features are wanted

Time: Best case this coming summer plus fall semester

End