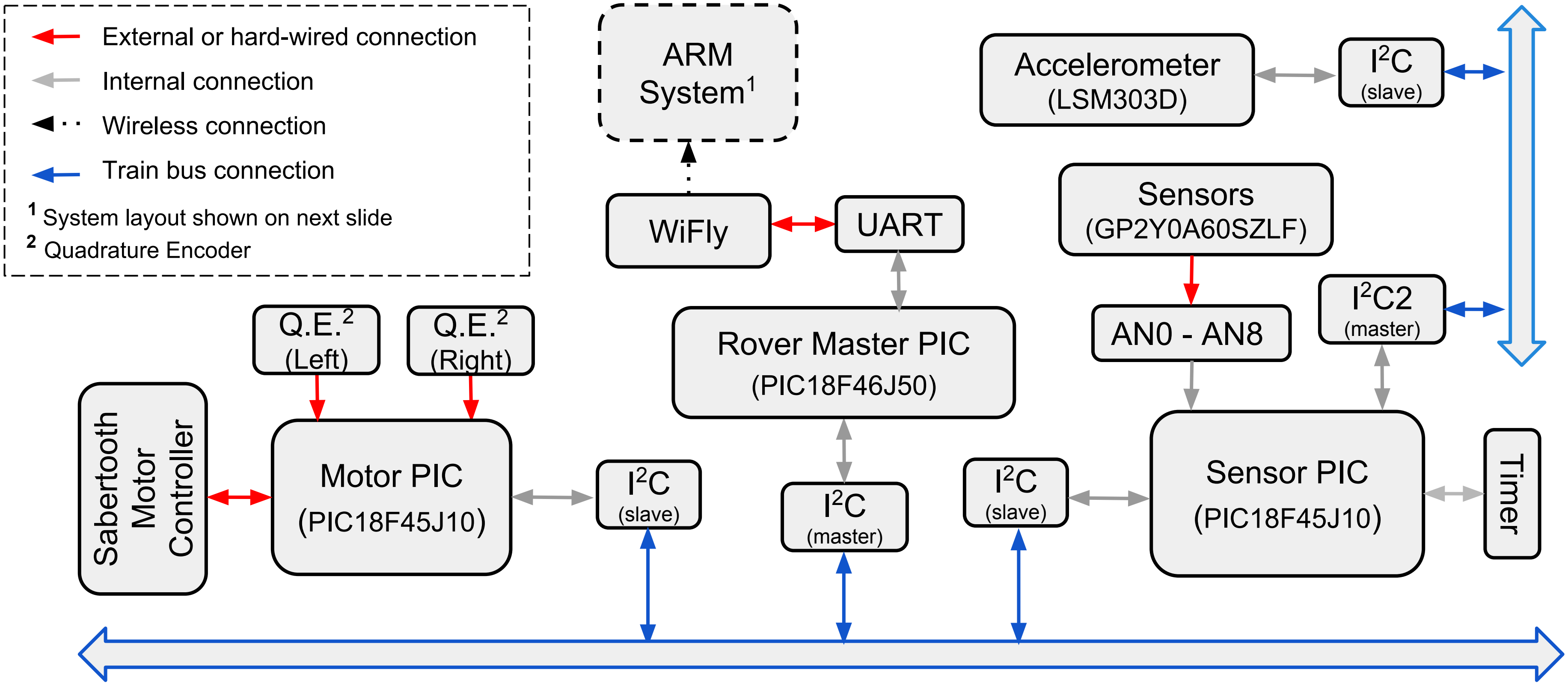
Embedded System Design

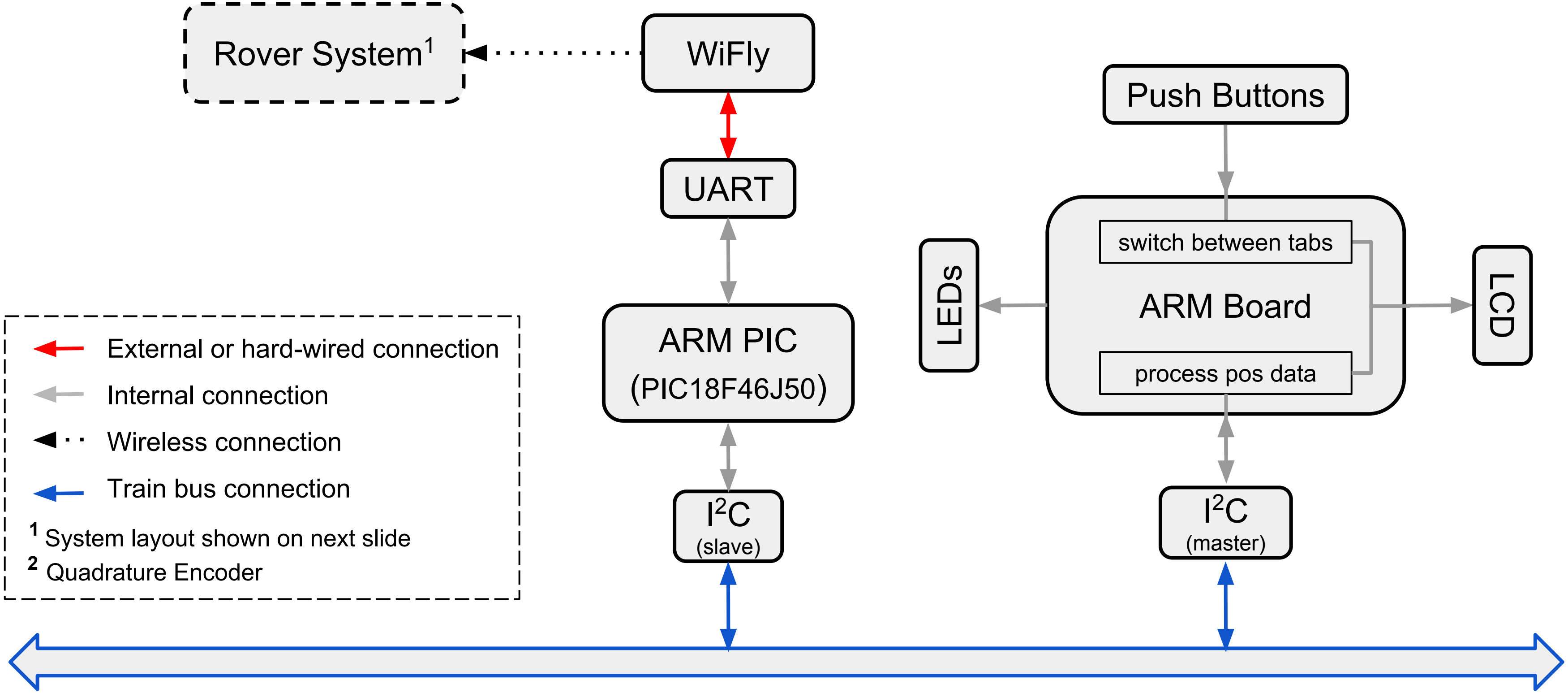
Project Overview

Igor Janjic
Leah Krynitsky
Brian Hilnbrand
Danny Duangphachanh

Component Communication



Component Communication Cont.



Device Communication Protocol

Message Overview

Source ⟶ Target

- Header: “H[Message type][Payload length]”
- Handshake: “X[Message Type]”
- Payload: “P[Payload]”
- Tail: “T”
- Handshake: “R[Size of payload received]”

Device Communication Protocol

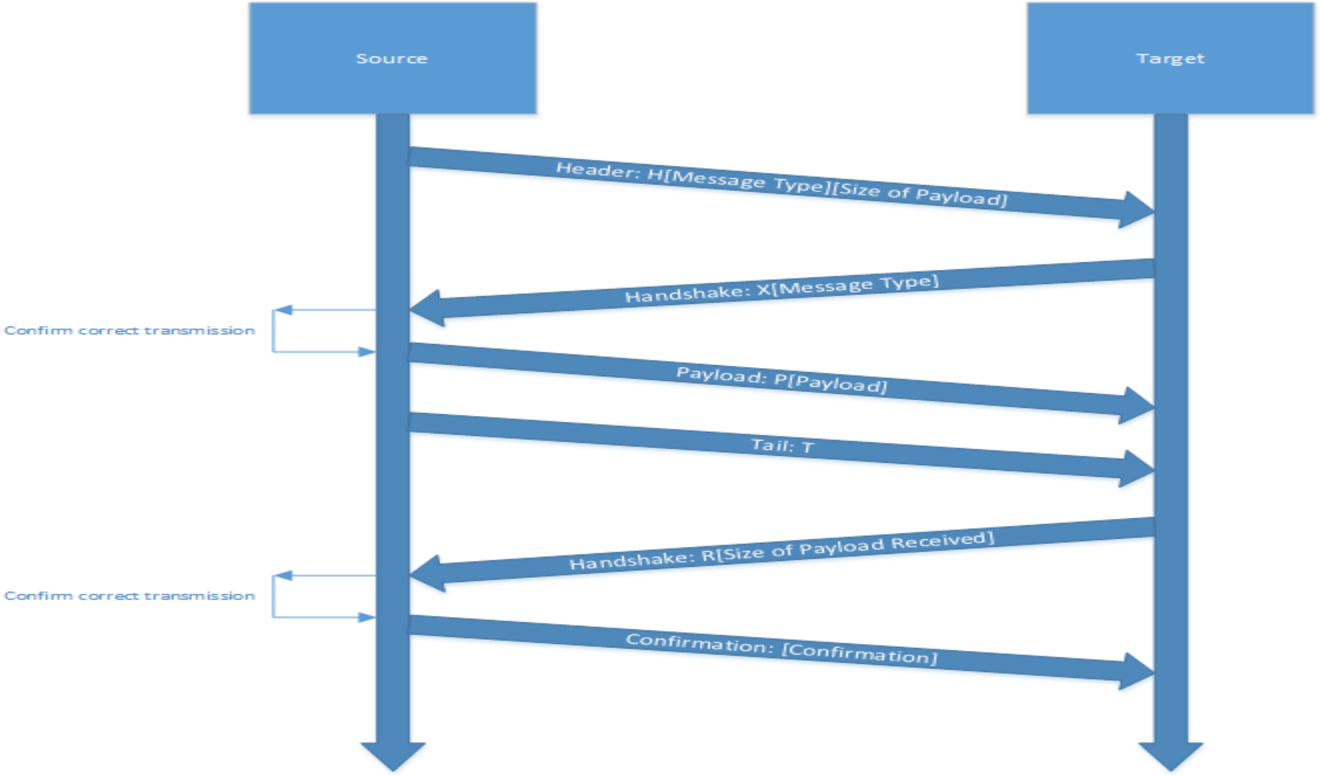
Message Types

Message Type	ID
MSGT_TIMER0	10
MSGT_TIMER1	11
MSGT_MAIN1	20
MSGT_OVERRUN	30
MSGT_UART_DATA	31
MSGT_I2C_DBG	41
MSGT_I2C_DATA	40

Message Type	ID
MSGT_I2C_RQST 42	42
MSGT_I2C_MASTER_SEND_COMPLETE	43
MSGT_I2C_MASTER_SEND_FAILED	44
MSGT_I2C_MASTER_RECV_COMPLETE	45
MSGT_I2C_MASTER_RECV_FAILED	46
MSGT_SENSOR_DATA	50

Device Communication Protocol

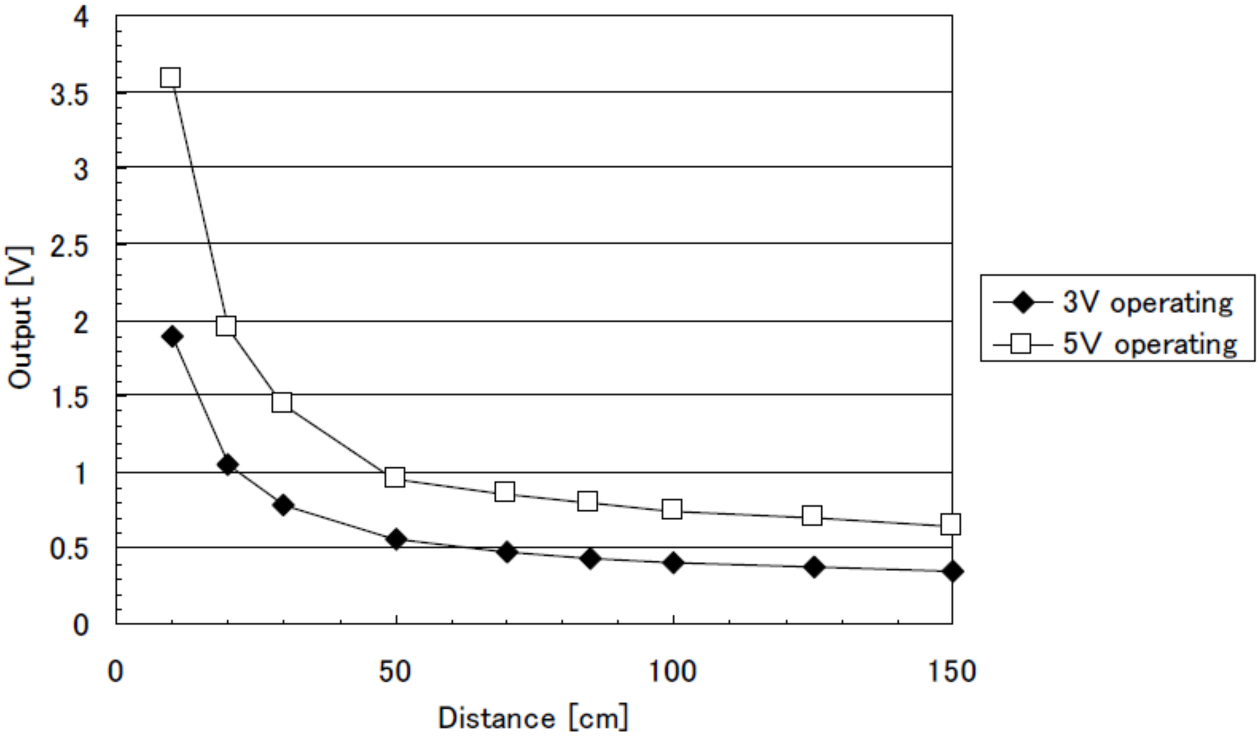
Message Flow



Sensor Selections & Specs

- SHARP GP2Y0A60SZLF Analog Distance Sensor
  - Measuring distance: 10 cm - 150 cm
  - Recommended operating voltage: 2.7 V - 5.5 V
  - Operating speed: 60 Hz (16.5 ± 3.7 ms update rate)
  - Output voltage differential: ~ 3.0 V (3.6 V at 10 cm - 0.6 V at 150 cm)
- LSM303D 3-Axis Accelerometer
  - Supply Voltage: 2.16 V - 3.6 V
  - I^2C interface
  - Operating speed: Anywhere from 0 kHz - 400 kHz
  - 16-bit data output

Sensor Analog Output vs. Distance



*Graph found in datasheet available at: http://www.pololu.com/file/0J812/gp2y0a60szxf_e.pdf

Sensor Locations on Rover

- Two IR sensors on each side of the rover for:
  - Object/obstacle detection
  - Direction calibration
  - General Orientation of Rover
- Two additional IR sensors on front side (center) spaced vertically for ramp detection
- One accelerometer on rover to detect rover tilt (flat surface or on ramp)

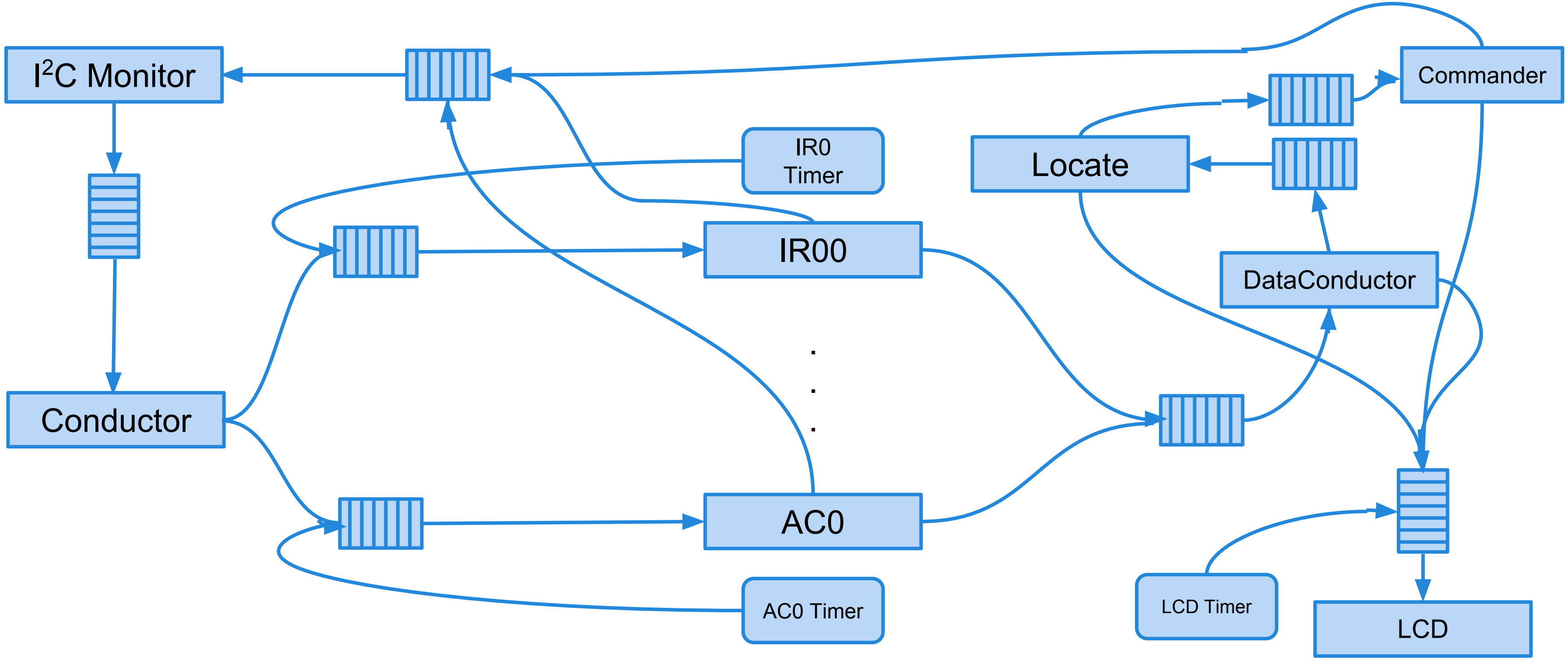
Sensor Communication

- IR Sensors hard wired to analog inputs of Sensor PIC (PIC18F45J10)
- Analog data from IR sensors converted using Sensor PIC A/D module
- Accelerometer connected via train bus to I^2C2
- Requests for data received via I^2C from Rover Master PIC (PIC18F46J50)
- 10-bit digital results from IR sensors and 16-bit results from accelerometer sent to Rover Master PIC via I^2C
- Data relayed to ARM PIC (PIC18F46J50) via WiFly and sent to ARM Board via I^2C

ARM Communication Overview

- ARM and 45J50 PIC (address 0x4F) communicate via I^2C
- Sensor queries sent to PIC 0x4F
- Sensor data received from PIC 0x4F
- Sensor data processed
- Sensor data displayed
- Rover and/or ramp locations determined
- Motor commands generated
- Motor commands sent to PIC 0x4F

ARM Tasks



I2C Monitor
Commander
IR0
Timer
Locate
IR00
DataConductor
. . .
Conductor
AC0
AC0 Timer
LCD Timer
LCD

ARM Communication
Sensor Data Protocol

- Sensor queries are sent to PIC 0x4F via I^2C
- Query rate is set as a compile time parameter
- For each query if no data is available, PIC 0x4F will reply with 0x00
- Received sensor data is preprocessed in various threads depending on which sensor the data corresponds to
- IR Sensors:
  - IR threads receive 2 bytes where the first byte is the first 8 bits of the sensor data and the last byte is the last 2 bits of the IR0 data
  - IR0 sensor located on the front of the rover spaced vertically
    - **IR00 thread** sends 0x0A to begin a query for data
    - **IR01 thread** sends 0x0B to begin a query for data
  - IR1 sensor located on the front of the rover spaced horizontally
    - **IR10 thread** sends 0x1A to begin a query for data
    - **IR11 thread** sends 0x1B to begin a query for data

ARM Communication

Sensor Data Protocol (Cont)

- IR Sensors:
  - IR threads receive 2 bytes where the first byte is the first 8 bits of the sensor data and the last byte is the last 2 bits of the IR0 data
  - IR2 sensor located on the right of the rover spaced horizontally
    - **IR20 thread** sends 0x2A to begin a query for data
    - **IR21 thread** sends 0x2B to begin a query for data
  - IR3 sensor located at the back of the rover spaced horizontally
    - **IR30 thread** sends 0x3A to begin a query for data
    - **IR31 thread** sends 0x3B to begin a query for data
  - IR4 sensor located on the left of the rover spaced horizontally
    - **IR40 thread** sends 0x4A to begin a query for data
    - **IR41 thread** sends 0x4B to begin a query for data
- **AC0 thread** sends 0x5A to begin a query for data from accelerometer sensor 0
  - AC0 thread receives 6 bytes (2 for each axis) in 2s compliment form

ARM Communication
Motor Command Protocol

- Motor commands sent to PIC 0x4F via I^2C
- **Commander thread** sends 1 byte representing the command that the motor controller will execute and 1 byte representing the speed the command will be executed at
- Then a query byte is sent that details whether or not the rover has hit an obstacle: 0x41:
  - No obstacle encountered: 0x00
  - Obstacle encountered: 0x01
- Commands are separated into two groups depending on the distance granularity needed: fine and coarse
- Fine commands:
  - Rotate left by 5 degrees: 0x11
  - Rotate right by 5 degrees: 0x12
  - Move forward 1 centimeters: 0x13
  - Move backward 1 centimeters: 0x13
- Coarse commands
  - Rotate left by 45 degrees: 0x21
  - Rotate right by 45 degrees: 0x22
  - Move forward 5 centimeters: 0x23
  - Move backward 5 centimeters: 0x24
- Speed:
  - Execute command at a slow rate: 0x31
  - Execute command at a medium rate: 0x32
  - Execute command at a fast rate: 0x33

ARM Algorithms
Motor Command

- Relevant object locations and the closest ramp location is sent to the **Command thread**
- Rover has four states:
  - **Roam**:
    - Rover moves until it discovers a ramp or encounters an obstacle
  - **RampDiscover**
    - Rover will periodically rotate 360 degrees to scan for ramps
  - **RampMove**
    - Rover moves toward the ramp
    - A sequential list of commands that get the rover to the ramp are kept in memory
    - List is updated with new data and newly discovered objects
  - **RampEscal**
    - Rover aligns with the ramp and goes over it

ARM Algorithms
Data Integration and Routing

- After incoming sensor data is received, it is sent to the **DataConductor thread** which further processes the data, integrates it, and routes it to the **Locate thread**, **LCD thread**, and **Commander thread**
- Routes data at a rate determined by compile time parameters
- Sends processed data to the LCD thread which displays it on appropriate tabs on the display
- Sends processed IR sensor data to Locate thread
- Sends processed AC sensor data to Commander thread
- Integration algorithms:
  - Filtering outliers
  - Converts IR sensor voltages to distances
  - Converts AC sensor voltages to angles

ARM Algorithms
Rover Localization

- After incoming IR sensor data is processed it is sent to the **Locate thread** which locates the rover on the internally supplied map, locates nearby ramps, and sends this data to the LCD
- Only data about relevant objects are sent to **Commander thread**
- Rover localization:
    - If an object is encountered it is appended to a list of currently known about objects
        - Distances to the object and counterclockwise angles of the object (with respect to the front of the rover) are kept in memory
        - List of currently known about objects are updated every time a new data point arrives.
        - Periodically, the list is scanned and collections of objects representing the same physical thing are integrated
    - The rover's location is determined by using this list

ARM Algorithms
Ramp Discovery

- Ramp Discovery:
  - Ramp discovery is performed using data from IR0 sensor
  - Only nearest ramp data is sent to **Commander thread**
  - If a ramp is discovered it is appended to a list of currently known about ramps
    - Distances to the ramp and counterclockwise angles of the ramp (with respect to the front of the rover) are kept in memory
    - List of currently known about ramps are updated every time a new data point arrives from the DataConductor thread

ARM Algorithms
Displaying Data

- DataConductor thread sends processed data to appropriate tabs on the LCD
- Tabs can be toggled with the pushbutton (or maybe by swiping the touch screen)
  - TAB0 displays the IR0 sensors voltages
  - TAB1 displays the IR1 sensors voltages
  - TAB2 displays the IR2 sensors voltages
  - TAB3 displays the IR3 sensors voltages
  - TAB4 displays the IR4 sensors voltages
  - TAB5 displays the AC0 sensor voltage
  - TAB6 displays a list of the distances to and angles from various objects the rover has encountered
  - TAB7 displays a live map of the rover.

Rover Master PIC Overview

- Rover Master PIC communicates with ARM via WiFly and Sensor PIC and Motor PIC via I^2C
- Receives sensor data request from ARM
- Reads sensor data message buffer for sensor data
- Transmits sensor data to ARM
- Receives motor commands from ARM
- Transmits motor commands to Motor PIC

Motor PIC Overview

- Motor PIC and Rover Master PIC communicate via I^2C
- Receives motor commands from ARM via Rover Master PIC
- Transmits motor commands as packets to Motor Controller on a single serial line
- Receives data from Quadrature Encoders to verify speed and turning angle

Motor Controller Operation

- Sabertooth 2x12 Motor Controller
- Receives motor command packet from Motor PIC
- Will operate in Mode 4: Packetized Serial
  - One-directional transmission
  - 9600 baud rate
  - Packet format consists of:
    - Address byte
    - Command byte
    - Data byte
    - 7-bit checksum

Serial Packet Details

- Address bytes are assigned by the DIP switch
- Will use four possible commands
  - Drive forward motor 1
  - Drive backwards motor 1
  - Drive forward motor 2
  - Drive backwards motor 2
- Data byte will determine the speed the motor will drive
  - 0 will have the motor go into power save mode
  - 127 will have the motor go full power
- If checksum is incorrect, the data packet will not be acted upon

Simulation Overview

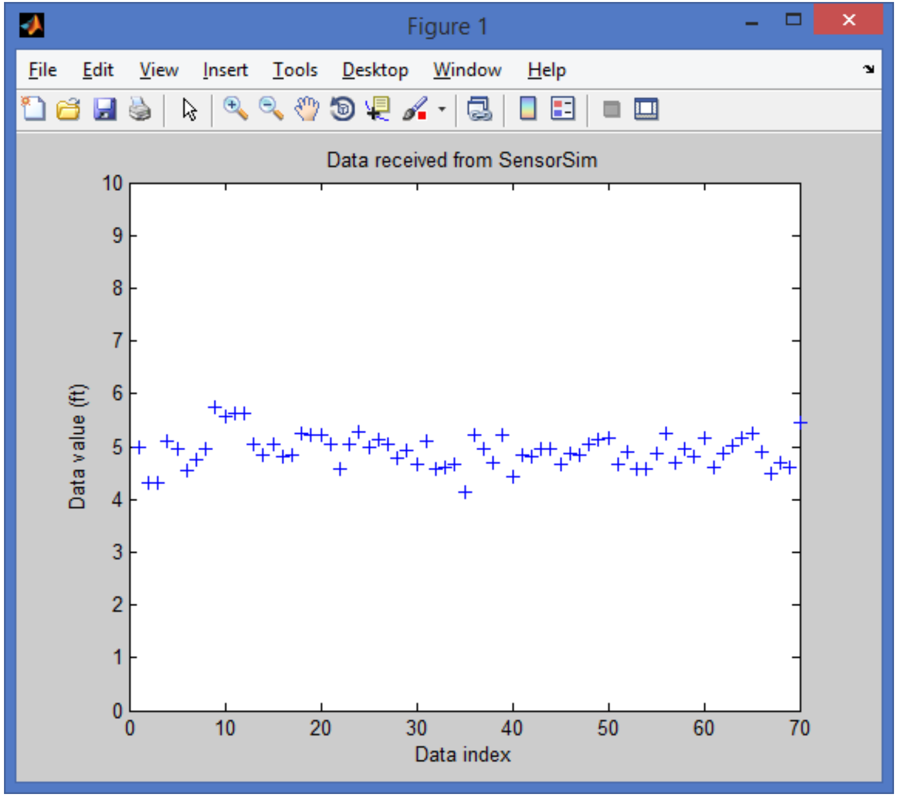
Why simulate?

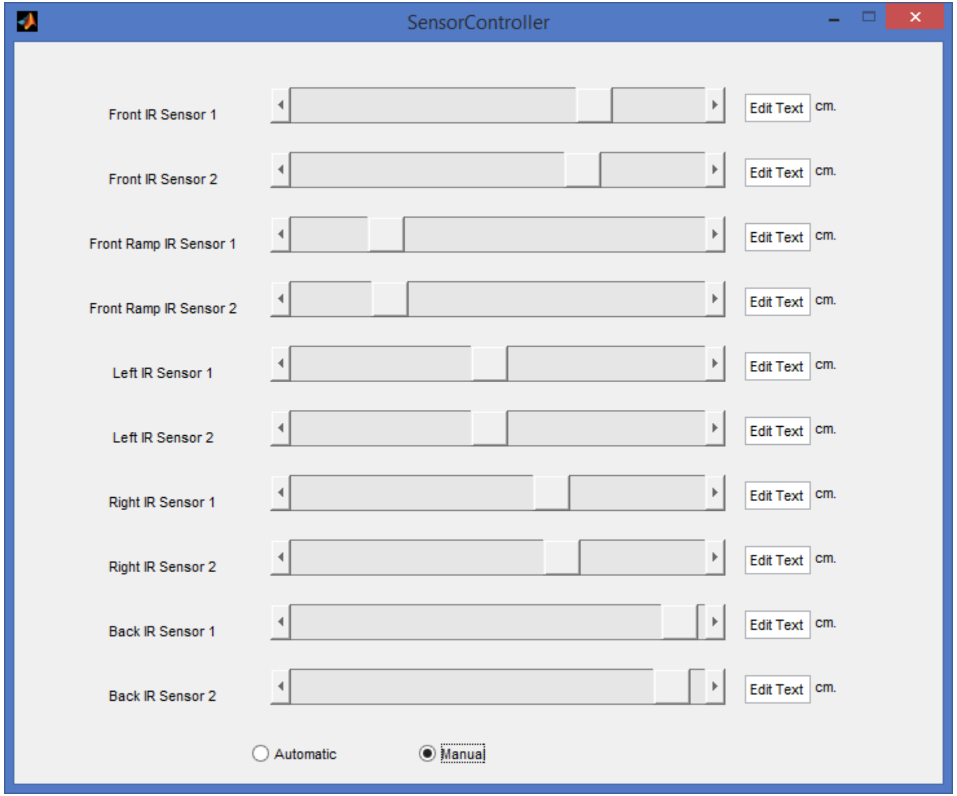
- Simulation is a key tool for engineers to test parts of products without having the entire product in-hand.
- Reduces development cost/time to a fraction of that of real testing.

What does it do?

- Mimics communication between component and the rest of the system.
- Does not actually emulate the rest of the system does, but generates fake data that could have been produced by the system.
- Presents data flow in a human-readable manner.

Simulation Overview





Questions?

#YOLOSWAG777BEARIT #3BARE4U #NOSCOPE #BEARSBEETSBATTLESTARGALACTICA #EXPLODINGBEAR

Team 7

Rover

Source: http://s3.amazonaws.com/wootdesigncontestentries/2141848/d77b5595-1fbc-4d5e-bcf3-baef36755aa2.jpg