TASK DIAGRAM

How to submit: Place a PDF document with the link/title "Task Diagrams" in your group's resources folder and make a link to this on your group's Wiki page. Your page must be readable on a web browser. What to submit: No more than 5 pages, formatted as below.

You are to describe your project design as a task graph in which the tasks (nodes of the graph) and the communication between tasks (edges of the graph) are clearly described. You need to clearly identify all sensor input and actuator output (e.g., motor controller) as nodes in the graph. At this stage of the design process, you do not have to identify any specific model numbers for components (e.g., IR sensor is enough, you don't need the specific model yet). Communication should be specified in terms of the type of data being sent and the approximate rate at which it is to be sent.

You are to describe your project design as a task graph in which the tasks (nodes of the graph) and the communication between tasks (edges of the graph) are clearly described. You need to clearly identify all sensor input and actuator output (e.g., motor controller) as nodes in the graph. At this stage of the design process, you do not have to identify any specific model numbers for components (e.g., IR sensor is enough, you don't need the specific model yet). Communication should be specified in terms of the type of data being sent and the approximate rate at which it is to be sent.

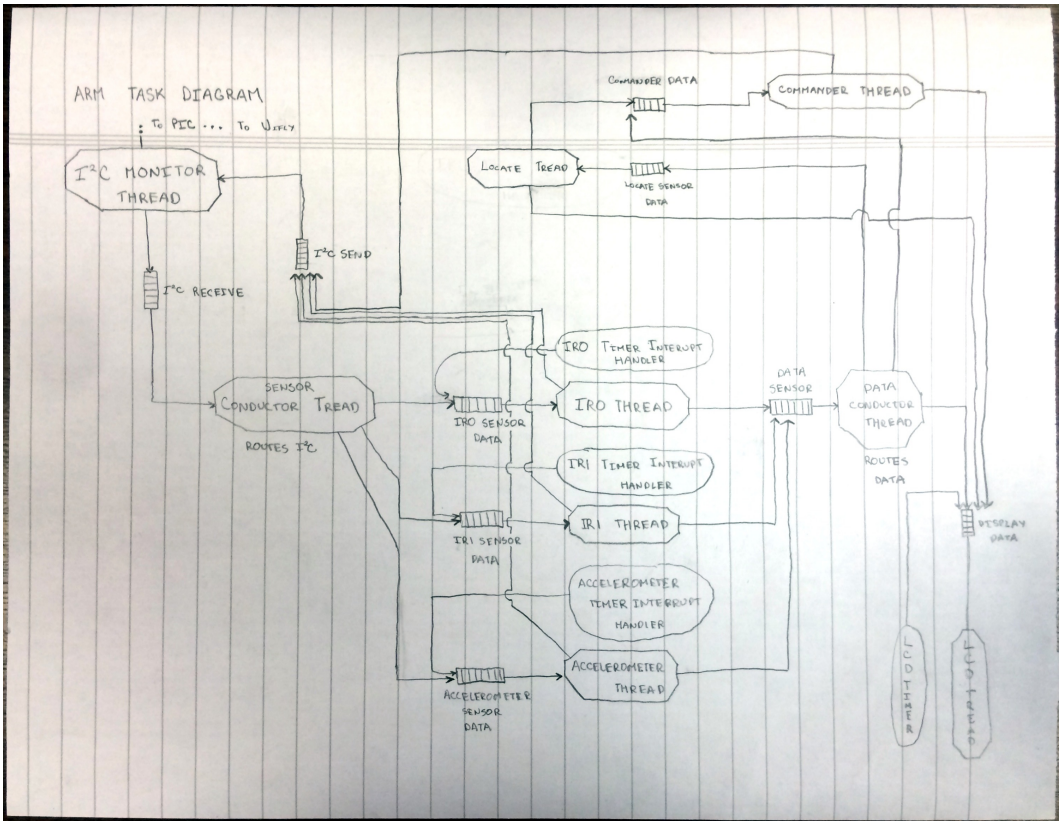
Created by:

Danny Duangphachanh
Leah Krynitsky
Brian Hilnbrand
Igor Janjic

September 18, 2014
[ECE 4534] Embedded Systems Design
Virginia Polytechnic Institute and State University

ARM Task Diagram

The following is the task diagram for the ARM board.



I²C Monitor Thread

1. This thread handles all I²C communications.
2. Implemented in `vtI2C.c`.
3. Queue: I²C Receive
   (a) This queue contains received messages via I²C and sends them to the Sensor Conductor Thread.
   (b) The received messages will have varying formats depending on the sensors that sent them. These formats will be described in the description for the sensor task diagram.
4. Queue: I²C Send
   (a) This queue contains sent messages via I²C to the master PIC.
   (b) The sent messages will have varying formats depending on the threads that send them.

Sensor Conductor Thread

1. This thread routes received I²C data and sends it to the various sensor threads.
2. Implemented in `sensorConductor.c`.

IR0 Thread

1. This thread processes received IR0 sensor data and sends it to the Data Conductor Thread.
2. Implemented in `i2cIR0.c`.
3. This thread has a timer interrupt handler.
4. Queue: IRO Sensor Data
   (a) This queue contains IR0 sensor data received via I^2C as well as data from the interrupt handler for this thread.
   (b) The received data will have type `unsigned int` and will represent the voltage of the IR0 sensor.

IR1 Thread

1. This thread processes received IR1 sensor data and sends it to the Data Conductor Thread.
2. Implemented in `i2cIR1.c`.
3. This thread has a timer interrupt handler.
4. Queue: IR1 Sensor Data
   (a) This queue contains IR1 sensor data received via I^2C as well as data from the interrupt handler for this thread.
   (b) The received data will have type `unsigned int` and will represent the voltage of the IR1 sensor.

Accelerometer Thread

1. This thread processes received accelerometer sensor data and sends it to the Data Conductor Thread.
2. Implemented in `i2cAccel.c`.
3. This thread has a timer interrupt handler.
4. Queue: Accelerometer Sensor Data
   (a) This queue contains accelerometer sensor data received via I^2C as well as data from the interrupt handler for this thread.
   (b) The received data will have type `int` and will represent the voltage of the accelerometer sensor.

Data Conductor Thread

1. This thread routes received sensor data from each sensor thread and sends it to a variety of processing threads. Relevant data is sent to the Locate Thread and LCD Thread.
2. Implemented in `dataConductor.c`.

3. Queue: Data Sensor
    (a) This queue contains all preprocessed sensor data.
    (b) The received data will have various types explained in each relevant sensor thread description.

LCD Thread

1. This thread displays the following on the LCD: raw sensor data (Data Conductor Thread), location data (Locate Thread), commands being sent to the motor controller (Command Thread).
2. Implemented in `LCDtask.c`.
3. This thread has a timer interrupt handler.
4. Queue: Display Data
    (a) This queue contains all data that will be displayed.
    (b) The received data will have various types explained in each relevant thread.

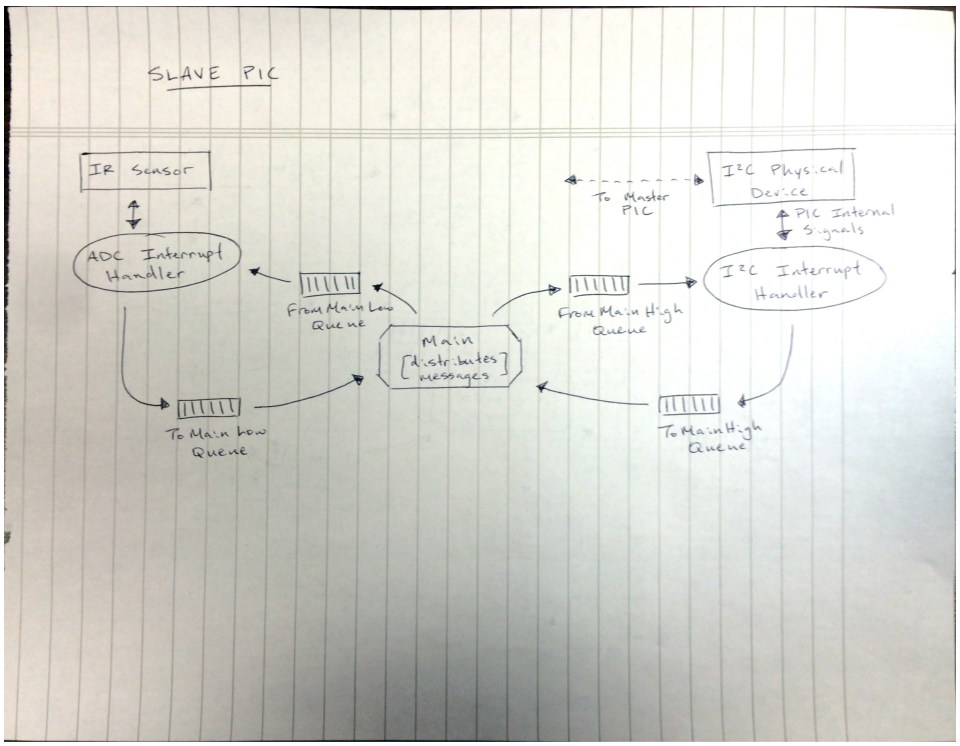
Locate Thread

1. This thread processes all relevant sensor data (IR0 and IR1) and determines the location of the rover which it then sends to the Commander Thread.
2. Implemented in `locateTask.c`.
3. Queue: Locate Sensor Data
    (a) This queue contains all sensor data used to find the location of the rover.
    (b) The received data will have various types explained in each relevant thread.

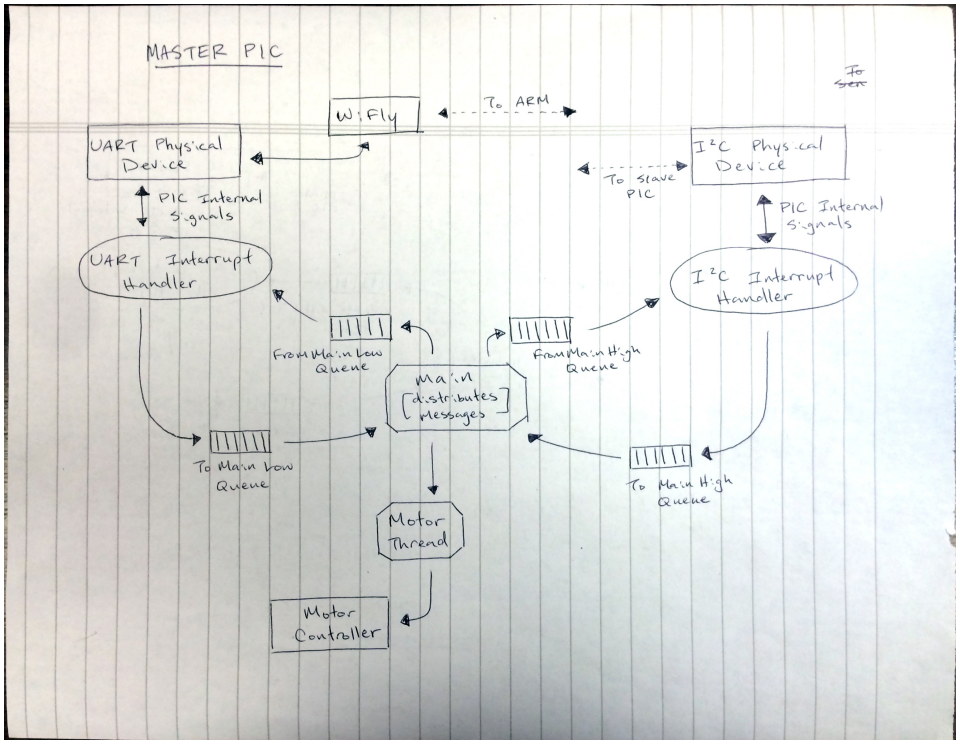
Commander Thread

1. This thread processes the location and determines the next motor movement which it then sends to the I^2 Moniter Thread (ultimately sending the command to the Motor PIC.
2. Implemented in `commanderTask.c`.
3. Queue: Commander Data
    (a) This queue contains all sensor data used to determine the next movement of the rover.
    (b) The received data will have various types explained in each relevant thread.

Slave Task Diagram



Master Task Diagram



Main Thread Distributes the message passing between MainLow and MainHigh queues and also handles initiation of all devices, threads, and interrupt handlers.

Motor Thread
Receives signals from Main. Communicates with the Motor Controller.

UART Interrupt Handler
Interrupts whenever the UART Physical Device is ready to send or receive a message.

I^{2}C Interrupt Handler
Interrupts whenever the I2C Physical Device is ready to send or receive a message to the other PIC.

ADC Interrupt Handler
Interrupts whenever the ADC is ready to initiate read or receive.

WiFly Physical Device
Wireless transmitter/receiver.

UART Physical Device
Communicates with the WiFly.

I^{2}C Physical Device
Communicates with the other I2C Physical Device to communicate with the other PIC.

Motor Controller Physical Device
Sends power to the motors.

IR Sensors Physical Devices
Reads the area directly in front of the sensor area.

ToMainHigh Message Queue
Handles messages from the I2C Physical Device to Main.

FromMainHigh Message Queue
Handles messages from Main to to I2C Physical Device.

ToMainLow Message Queue
Handles messages from the UART and IR Sensor on the Master and Slave PIC, respectively.

FromMainLow Message Queue
Handles messages from Main to the UART and IR Sensor on the Master and Slave PIC, respectively.