The Synapse UAV is an unmanned autonomous vehicle that is composed of several devices. The motivation behind this project is to learn about autonomous vehicles guided by GPS technology.
VEHICLE
The Synapse UAV control system will be designed so any RF controlled vehicle can be used by loading a compatible software package onto the PocketPC. Our implementation vehicle will be an RF remote controlled car with forward/back and right/left controls.

Tasks:
- Determine how the motor system will interface with guidance controller
- Define a software API flexible enough to admit several motor control systems.
The team has already begun to test methods of integrating the RF controllers that come with the vehicles into the rest of the system. The integration should be done with little or no permanent modification of the vehicle or its controller.
Vehicle
GUIDANCE CONTROLLER
The guidance controller interfaces between the motors and the Pocket PC. It takes in a serial input from the Pocket PC, interprets the data, and sends out signals to the vehicle’s motors. The signal to the motors could either be an RF signal to take advantage of the current motor setup or we could bypass the RF system and send an analog signal directly to the motors. The guidance controller will be composed of a Motorola M9S12C32 microcontroller with an integrated serial interface. The M9S12C32 was provided free of charge by FreeScale Corporation.

Tasks:
- Combine serial interface and microcontroller and come up with messaging standard.
Guidance Controller

**SRFo8 – Devantech**

High Performance Ultrasonic Range Finder.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I²C Protocol</td>
<td></td>
</tr>
<tr>
<td>65mS</td>
<td></td>
</tr>
<tr>
<td>6 meter range (up to 11 meters)</td>
<td></td>
</tr>
<tr>
<td>Light Sensor</td>
<td></td>
</tr>
</tbody>
</table>
**Guidance Controller**

**I²C Communication Protocol**

**Steps**

1. Send start info and begin clock on SCL
2. Send address a certain slave device (sonar)
3. Send bit for send or receive and get an acknowledge back
4. Send/receive 8 bit data on SDA and get an acknowledge back
5. Go to step 3 or send end info
Guidance Controller

Steps

1. To initiate a ranging, the microcontroller sends a command 0x51.
2. The microcontroller sends a read to the sonar and will repeatedly read 0xFF until the sonar is done ranging and sends out the distance in centimeters.
3. The microcontroller reads the data and stores it another location for other parts of the microcontroller to use.
4. Go to 1.

Input & Output Sonar Bits

<table>
<thead>
<tr>
<th>Location</th>
<th>Read</th>
<th>Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Software Revision</td>
<td>Command Register</td>
</tr>
<tr>
<td>1</td>
<td>Light Sensor</td>
<td>Max Gain Register (default 31)</td>
</tr>
<tr>
<td>2</td>
<td>1st Echo High Byte</td>
<td>Range Register (default 255)</td>
</tr>
<tr>
<td>3</td>
<td>1st Echo Low Byte</td>
<td>N/A</td>
</tr>
<tr>
<td>~~~~</td>
<td>~~~~</td>
<td>~~~~</td>
</tr>
<tr>
<td>34</td>
<td>17th Echo High Byte</td>
<td>N/A</td>
</tr>
<tr>
<td>35</td>
<td>17th Echo Low Byte</td>
<td>N/A</td>
</tr>
</tbody>
</table>
POCKET PC & GPS UNIT
Pocket PC & GPS Unit

The Pocket PC is used to make calculations based on a predefined path by using the data from the attached GPS unit. It will not only store the main program, but also the travel path of the UAV. The SD port will be used to connect the GPS unit to the Pocket PC and also provide power to it. Once it receives this data, the Pocket PC will then be able to calculate the path that the UAV must travel. It will relay this information to the guidance controller via serial interface.
The GPS data will be parsed with the use of GeoFrameworks GPS utilities. We will use Visual Studio and C# to write all of the Pocket PC software.

The Pocket PC comes from one of the group members.

The GPS unit has been procured through an online vendor.

Tasks:

- Write main application to calculate direction based on GPS information

```csharp
// Creates a new instance using the specified value and unit type.
Speed Speed1 = new Speed(
    // value (System.Double)
    1.0,
    // units (SpeedUnit)
    SpeedUnit2);

// Create a new Speed object.
Speed Speed2 = new Speed(
    // value (System.String)
    "value");
```
Pocket PC & GPS Unit

$GPRMC,POS_UTC,POS_STAT,LAT,LAT_REF,LON,LON_REF,SPD,HDG,DATE,MAG_VAR,MAG_REF*CC<cr><lf>
$GPGLL,LAT,LAT_REF,LONG,LONG_REF,POS_UTC,POS_STAT*CC<cr><lf>
$GPGGA,POS_UTC,LAT,LAT_REF,LONG,LONG_REF,FIX_MODE,SAT_USED,HDOP,ALT,ALT_UNIT,GEO,G_UNIT,D_AGE,D_REF*CC<cr><lf>

POS_UTC - UTC of position. Hours, minutes and seconds [fraction (opt.)]. (hhmmss.[fff])
POS_STAT - Position status. (A = Data valid, V = Data invalid)
LAT   - Latitude (llll.ll)
LAT_REF - Latitude direction. (N = North, S = South)
LON   - Longitude (yyyyy.yy)
LON_REF - Longitude direction (E = East, W = West)
SPD   - Speed over ground. (knots) (x.x)
HDG   - Heading/track made good (degrees True) (x.x)
DATE  - Date (ddmmyy)
MAG_VAR - Magnetic variation (degrees) (x.x)
MAG_REF - Magnetic variation (E = East, W = West)
FIX_MODE - Position Fix Mode ( 0 = Invalid, >0 = Valid)
SAT_USED - Number Satellites used in solution
HDOP   - Horizontal Dilution of Precision
BASE STATION & CAMERA
The base station software will also be written in C# and run from a laptop. The travel path of the UAV will be displayed on the base station along with any other relevant data, such as GPS coordinates, altitude, etc. An RF camera on the vehicle with an RF receiver on the laptop is also being considered.

The RF receiver for the laptop, along with the camera to be mounted onto the vehicle, will need to be purchased.

Tasks:

- Write base station application with video display module from RF receiver.