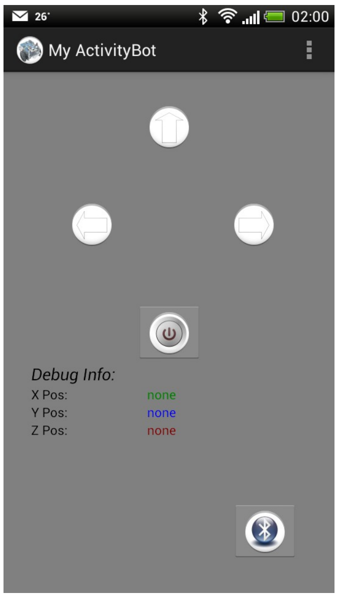
MyActivityBot

Motion Control Your Parallax ActivityBot



“Hope you'll have more fun with your bot!”

Kenichi K.
Designer & Developer

Getting Started

What you'll need:

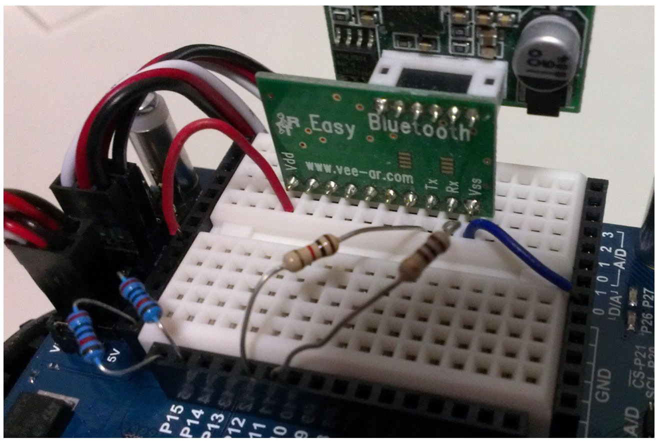
- Parallax ActivityBot
- *Bluetooth module (mount on ActivityBot)
- Android smartphone (Android OS 2.2 & above)

*Please ensure your Bluetooth module has UART (Rx/Tx) connectivity to your Propeller on your ActivityBot. In this example, I'm using Parallax's Easy Bluetooth module, which as of this writing, is a discontinued product.

Setup

ActivityBot

1. Please assemble & calibrate your ActivityBot according to Parallax's manual & documentation.
2. Mount your Bluetooth module onto the mini breadboard on the Activity board. In my example & code, Bluetooth's Rx connects to P10 via a 100ohm resistor in series & Tx connects to P11 via a 1Kohm as current-limiting resistor. Then connect the Vdd & Vss accordingly.

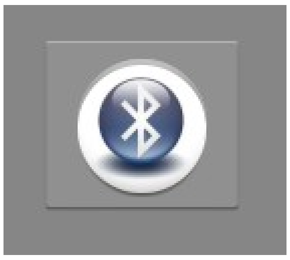


3. Launch Parallax SimpleIDE & load the MyActivityBot C code into your bot's EEPROM.
4. Switch OFF your ActivityBot.

Launch & Play

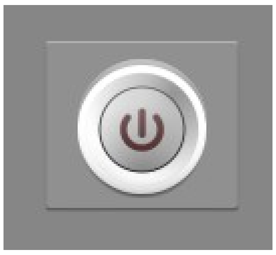
Android Smart Phone

1. Launch Google Play app & search for "MyActivityBot".
2. Install the app.
3. Once installed, launch the app.
4. If your Bluetooth is not currently turned on, it will ask you to switch ON.
5. Press the Bluetooth icon & select your paired Bluetooth module. If not paired, press "Scan for devices" to search & add to your phone.



6. Once done, switch on your ActivityBot (switch# 2) & turn your phone pointing it downwards to ensure zero acceleration.

7. Then, press the Power Switch icon to start the communication with your ActivityBot.



X-axis controls Left & Right

Y-axis controls Acceleration

Pointing Down => Slow/Stop

Pointing Up => Fast

C code (PropGCC)

MyActivityBot.c

```
/*
  Project: Controlling MyActivityBot via Bluetooth/
  Author: Kenichi K. (a.k.a. MacTuxLin)

*/
#include "simpletools.h"                    // Include simple tools
#include "KenichiActivityBot.h"
#include "fdserial.h"
#include "abdrive.h"

int *portNumPt;   // Handler for UART with Bluetooth
volatile unsigned int statusFlag = false;   // User's switch
uint  stack[(160 + (50 * 4)) / 4];

int main(){                                  // Main function

  // Init
  int   cogLED;
  int   rxData;
  int   cmdStream[BUFFLENGTH];

  // Setup
  //hwSetup();
  //hwSetup(rxPort, txPort, modeValue, baudRate);
  fdserial *portNumPt = fdserial_open(rxPort, txPort, modeValue, baudRate);
  pause(500);

  // Launch Cog# 1
  cogLED = cogstart(&connectionLED, NULL, stack, sizeof(stack));

  while(1){
```

```
// Get Cmd
//rxData = fdserial_rxCheck(portNumPt);
rxData = fdserial_rxChar(portNumPt);
rxData ^= CHECKSUM;

//print("rxData : %d", rxData);

// Checking for Start/Stop Byte
switch(rxData){
  case STARTBYTE:
    statusFlag = true;
    drive_trimSet(0, 0, 0);
    break;
  case STOPBYTE:
    statusFlag = false;
    drive_speed(0, 0);
    drive_trimSet(0, 0, 0);
    break;
}

// Processing Cmd
if(statusFlag){
  // Get set stream of data
  while(fdserial_rxCheck(portNumPt) != CHECKSUM){}
  //dummy = fdserial_rxChar(portNumPt);
  for(int i=0; i<3; i++){
    cmdStream[i] = fdserial_rxChar(portNumPt);
  }

  //--- Move ActivityBot ---

  // Debugging: Testing
  print("\nL: %d | ", cmdStream[0]);    // Left Speed
  print("R: %d | ", cmdStream[1]);      // Right Speed
  print("F: %d ", cmdStream[2]);        // Speed Multiplier
```

```
        // Debugging: Testing

        drive_speed(cmdStream[0] * cmdStream[2], cmdStream[1] * cmdStream[2]);
        //pause(50);
      }else{
        drive_speed(0, 0);
      }

    }
  }


  void hwSetup(int rx, int tx, int mode, int baud){
  //void hwSetup();
    fdserial *portNumPt = fdserial_open(rx, tx, mode, baud);
    //fdserial *portNumPt = fdserial_open(rxPort, txPort, modeValue, baudRate);
    pause(500);
  }


  void connectionLED(void *par){

    int delayDuration = 0;

    while(1){
      if(statusFlag==0){              //<-- Not consistent, not sure why???
        //delayDuration = notConnected;
        delayDuration = 1000;
      }else{
        //delayDuration = connected;
        delayDuration = 150;
      }

      high(signalLED);
      pause(delayDuration);
      low(signalLED);
```

```
        pause(delayDuration);
      }
    }
```

MyActivityBot.h

```
// H/W declaration
  // Drives
#define leftServo     12
#define rightServo    13
#define leftEncoder   14
#define rightEncoder  15

  // Feedback
#define signalLED     26


  // Bluetooth/UART
#define rxPort        11
#define txPort        10
#define modeValue     0
#define baudRate      9600

// Delay
#define signalPause   100
#define notConnected  1000
#define connected     150

// Logics
#define true          1
#define false         0

// Cons
#define uint          unsigned int

// Comm Protocol
#define STARTBYTE     0xA1
#define STOPBYTE      0xAF
#define CMDSTREAM     0xAA
#define CHECKSUM      0x7F
#define BUFFLENGTH    6



// Function Prototypes
//void hwSetup();
void hwSetup(int rx, int tx, int mode, int baud);
void connectionLED(void *par);  // Cog 1
```