Introduction to *Roomba* and *Create*

Jamie Snape

The University of North Carolina at Chapel Hill
Vacuum cleaning robots

- Earliest related patents issued in 1970’s for autonomous lawn mowers
- Hitachi and Sanyo demonstrated prototypes in 1991
- First commercial product released in 2000 by Electrolux of Sweden
Roomba

- First released in 2002 and patented 2005
- Development took 30 prototypes over twelve years
- Two-millionth Roomba sold in 2006
- Now in its fifth generation, the 500 series

© iRobot Corporation
Roomba actuators

- Two drive wheels
- Agitator brush
- Side brush
- Vacuum
- (Speaker)
Roomba steering

- Differential steering
  - Each wheel controlled by a distinct motor
  - Allows Roomba to spin on the spot
Roomba sensors

- Over 30 sensors including:
  - Bump and wheel drop (mechanical)
  - Wall and virtual wall (infrared)
  - Wheel overcurrents (electrical)
  - Dirt detector (piezoelectric)
Roomba heuristics

• Spiraling: Cleans a concentrated area

• Wall following: Follows the perimeter of the room

• Room crossing: Crisscrosses the room to achieve full coverage

• Dirt detection: Senses dirt and cleans more intensively
Human-Roomba interaction

- Georgia Tech interviewed Roomba owners:
  - Most named their Roomba and decided on a gender
  - Many “worried” or “felt sorry” for Roomba if it became stuck
  - “Roombarization” to help and protect Roomba

© My Room Bud
Based on the *Roomba* platform

- *Scooba*: Floor washing robot
- *Dirt Dog*: Workshop sweeping robot
- *ConnectR*: “Virtual visiting” robot
- *Create*: Programmable robot

© iRobot Corporation
Create

- Almost identical to fourth generation *Roomba* except:
  - No vacuum
  - Extra LEDs and louder speaker
  - Serial port for adding extra actuators, sensors, communications, microprocessor, etc.
Programming *Create*

- Access actuators and 32 sensors over serial port
- *Create* accepts serial commands and sends back sensor packets
- Connect via serial cable or Bluetooth and use any scripting language
- Attach a microcontroller and use a subset of C

© Acroname Robotics
Create Open Interface

• Electronic interface: 7-pin Mini-DIN (also present on Roomba) and DB-25 connectors

• Software interface: Mode, actuator, song, demo, sensor commands, etc.

• Each command is a one-byte opcode, possibly followed by data bytes
Open Interface modes

- Passive: Request and receive sensor data only
- Safe: Control of actuators, but if a cliff or wheel drop is detected, or the charger is attached, switches to passive mode
- Full: Complete control of actuators, all safety features turned off
- Opcodes 128, 131, and 132
Example

- Start *Create*, switch to full mode, run some commands, then switch to passive:
- [128] [132] ... [128]
- *Create* always starts in passive mode
Driving *Create*

- Opcode 137 with four data bytes interpreted as two 16-bit signed values
- First two bytes velocity, second two radius
- Special cases for radius:
  - Straight = hex 8000 or 7FFF
  - Clockwise spin = hex FFFF
  - Counterclockwise spin = hex 0001
Example

- Reverse at a velocity of -200 mm/s while turning at a radius of 500 mm:
  - $\text{Velocity} = -200 = \text{hex FF38} = \text{hex FF hex 38} = 255 \ 56$
  - $\text{Radius} = 500 = \text{hex 01F4} = \text{hex 01 hex F4} = 1 \ 244$
Reading sensors

- *Create* updates its sensors every 15 ms
- 43 sensor packets representing a sensor or group of sensors
- Use opcode 142 with one data byte to request packets
- Special packet IDs:
  - Specific subgroups: 0 - 5
  - All sensors: 6
Example

• Get the state of the left cliff sensor:
• [142] [9]
  • Cliff sensors are packets 9 - 13
Command Module

- Plugs into cargo bay connector
- Four expansion ports for custom hardware
- Powered by Atmel AVR ATMega168 microcontroller
- 8-bit RISC architecture with ~18 MHz crystal, 14336 bytes flash memory
AVR-GCC Toolchain
Command Module programming

- Main software components:
  - `avr-gcc`: C compiler
  - `avrdude`: Programmer
- Possible to program in C++, but poorly supported
- Easiest to adapt and extend example code and makefiles
- `oi.h` contains Open Interface definitions
#include "oi.h"
#include <avr/interrupt.h>
#include <avr/io.h>

void initialize(void) {
    cli(); // Turn off interrupts

    // Set I/O pins
    DDRB = 0x10;  PORTB = 0xCF;  DDRC = 0x00;
    PORTC = 0xFF;  DDRD = 0xE6;  PORTD = 0x7D;

    // Set baud rate to 57600 bps
    UBRR0 = 19;  UCSR0B = 0x18;  UCSR0C = 0x06;

    // Set up timer 1 to generate an interrupt every 1 ms
    TCCR1A = 0x00;
    TCCR1B = (_BV(WGM12) | _BV(CS12));
    OCR1A = 71;
    TIMSK1 = _BV(OCIE1A);

    sei(); // Turn on interrupts
}
```c
#define Sen6Size 52
#define CmdSensors 142

volatile uint8_t sensors_flag = 0;
volatile uint8_t sensors_index = 0;
volatile uint8_t sensors_in[Sen6Size];
volatile uint8_t sensors[Sen6Size];

if(!sensors_flag) {
    for(temp = 0; temp < Sen6Size; temp++)
        sensors[temp] = sensors_in[temp];
    byteTx(CmdSensors); byteTx(6);
    sensors_index = 0; sensors_flag = 1;
}

// Serial receive interrupt to store sensor values
ISR(USART_RX_vect) {
    if(sensors_flag) {
        sensors_in[sensors_index++] = UDR0;

        if(sensors_index >= Sen6Size)
            sensors_flag = 0;
    }
}
```
Timer interrupt

volatile uint16_t timer_cnt = 0;
volatile uint8_t timer_on = 0;

// Timer 1 interrupt to time delays in ms
ISR(TIMER1_COMPA_vect) {
    if(timer_cnt > 0)
        timer_cnt--;
    else
        timer_on = 0;
}

Send and receive

// Transmit a byte over the serial port
void byteTx(uint8_t value) {
    while(!(UCSR0A & 0x20)); // Do nothing
    UDR0 = value;
}

// Receive a byte from the serial port
uint8_t byteRx(void) {
    while(!(UCSR0A & 0x80)); // Do nothing

    return UDR0;
}
# Drive function

```c
#define CmdDrive 137

// Send Create drive commands in terms of velocity and radius
void drive(int16_t velocity, int16_t radius)
{
    byteTx(CmdDrive);
    byteTx((uint8_t)((velocity >> 8) & 0x00FF));
    byteTx((uint8_t)(velocity & 0x00FF));
    byteTx((uint8_t)((radius >> 8) & 0x00FF));
    byteTx((uint8_t)(radius & 0x00FF));
}
```
Other options

- *Create* and *Roomba* can be controlled by these platforms via serial cable or Bluetooth and/or simulated (in part):
  - Cogmation robotSuite
  - Microsoft Robotics Developer Studio
  - Player/Stage/Gazebo
References

References

• iRobot Corporation (2007): iRobot Create Open Interface Specification

• iRobot Corporation (2007): iRobot Roomba 500 Series Owner’s Guide

• Jones et al. (2005): Autonomous floor-cleaning robot. *U.S. Patent 6,883,201*
References

- Jones et al. (2004): Method and system for multi-mode coverage for an autonomous robot. *U.S. Patent 6,809,490*


- Sung et al. (2007): “My Roomba is Rambo”: Intimate home appliances. *UbiComp 2007*