Remote Programmable Wake-Up Alarm

Ian Mowers
ECE-499 Design Presentation

Introduction

Problem:

- Shared room situation
- Alarms that wake others aside from the user
- Quieter wake up process

What Others have done:

- Two similar products in development
- Both use in-ear receivers
- Separate timing device
- Set by smartphone

Goals

- Design a quiet wake-up alarm that is controlled by smartphone
- Personal device, only wakes user
- User can set or cancel alarm
- Automatically activates

Criteria

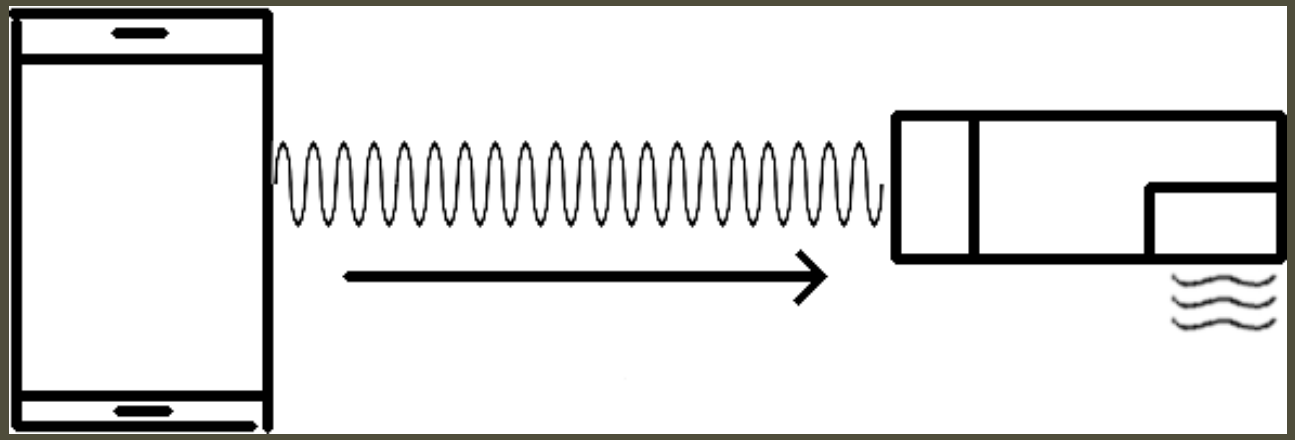
- Wireless, Range: 20x20m
- Size: 3-7cm length
- Quiet enough to avoid waking others
- Loud enough to wake user
- Battery Usage: Several days
- Rechargeable (Potential)

Overall Design

Smartphone

Communication

Alarm Receiver

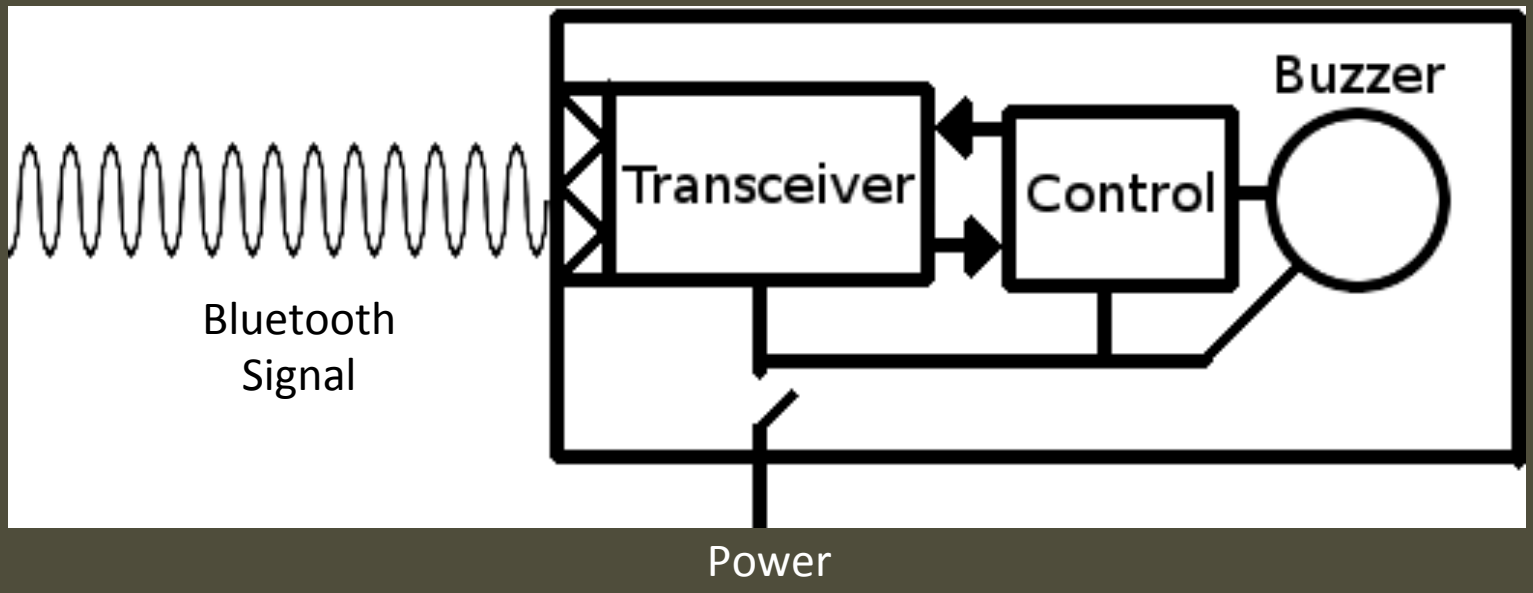


Application
Acts as Transmitter

One Way
Bluetooth

Receive Signal
Enable
Vibration
Requires Power

Alarm Design

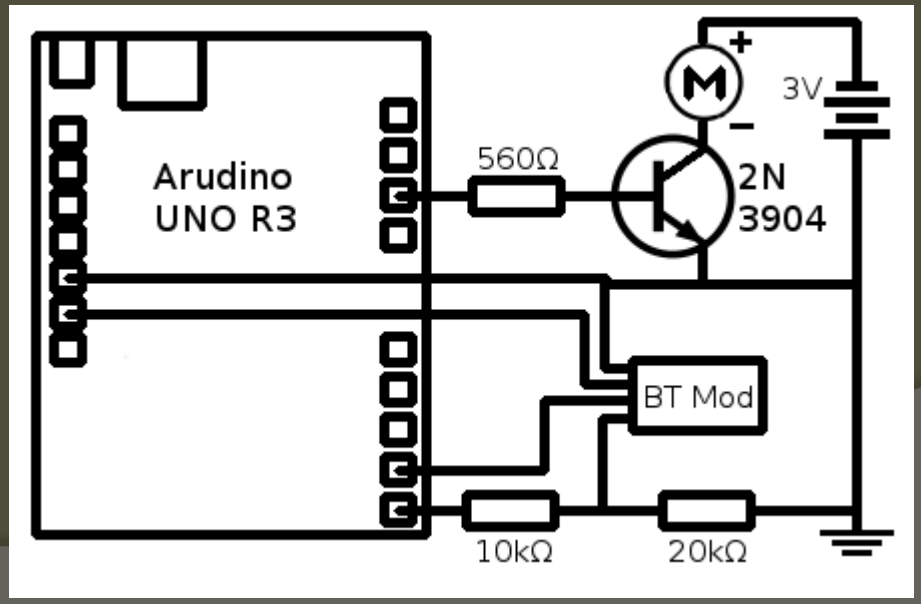


Overall Implementation

- Smartphone:
  - Android OS
  - Minimum SDK 19 or later "Kit-Kat" and newer
  - Covers most Android phones
  - Built-in Bluetooth
  - Utilize System Alarm Service
- Alarm:
  - Bluetooth Transceiver
  - Vibration Motor
  - Control -
    - Arduino UNO R3
    - PIC24FV32KA304

Receiver Implementation

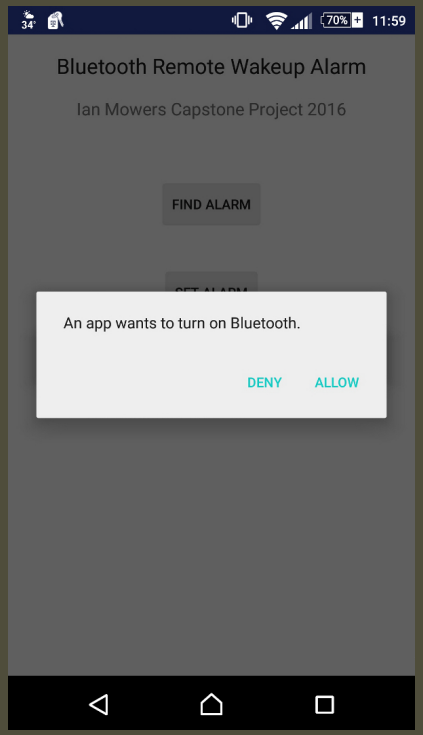
- Arduino-based:
  - Power transceiver with Arduino
  - Power motor from 3V button battery
  - Voltage divider for transceiver Rx pin
- PIC-based:
  - Power - 2-3V button batteries
  - Special pin set up
  - Voltage regulator to 5V
  - Voltage dividers for all I/O pins

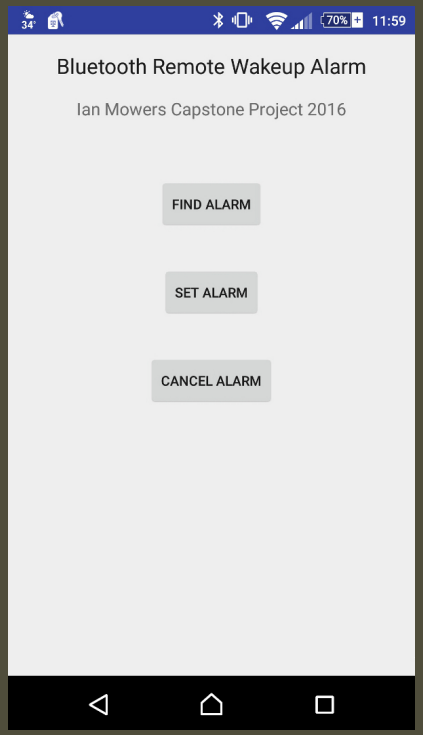


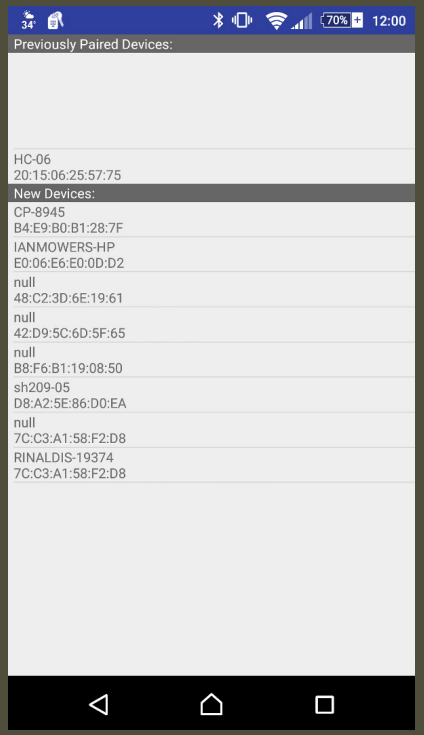
Receiver Logic

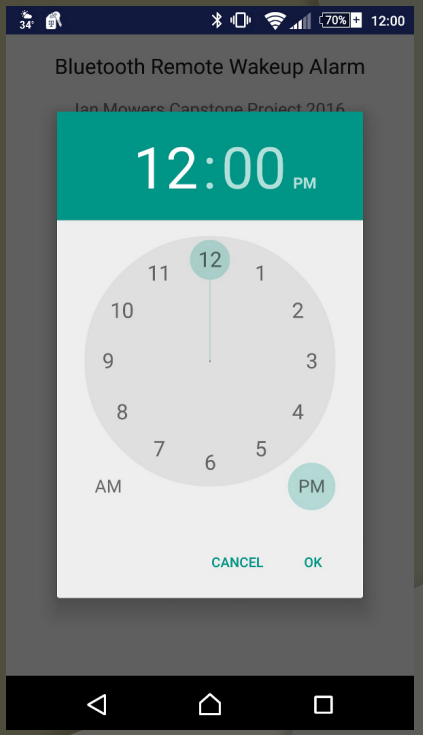
- Arduino-based:
  - Set motor output pin
  - Open Serial read
  - Wait until receive
  - Read input, check if ‘1’
  - Turn on motor
  - Wait maximum allowable time
  - Turn off motor
- PIC-based:
  - Set up UART Module
  - Wait for Interrupt
  - Check Rx Register
  - Check for parity and overflow
  - Check if value is a ‘1’
  - Turn on I/O pin
  - If time limit reached, turn off pin

Smartphone Application



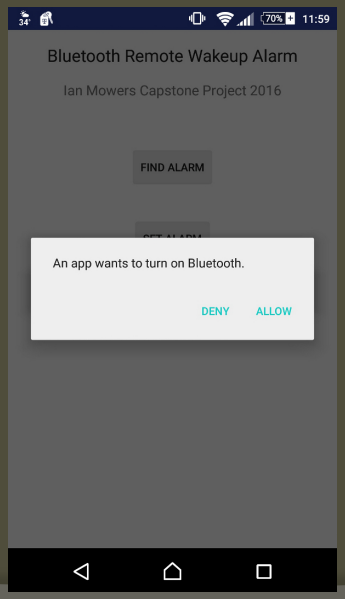


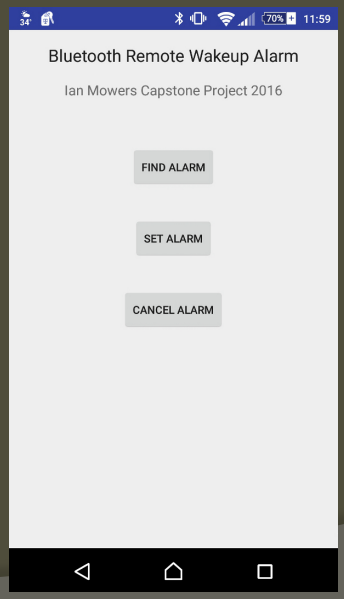


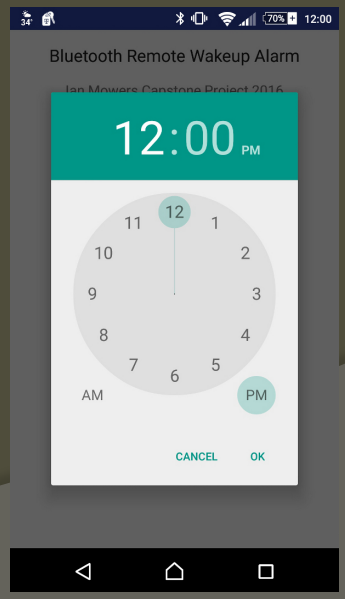


Smartphone Application

- Main Activity
  - Serves as main menu
  - Maintains TimePicker dialog
  - Where user sets alarm
  - Allows user to cancel alarm
  - Access system alarm service
- Important Built-in Classes:
  - TimePicker
  - AlarmManager*
  - java.util.Calendar
  - WakefulBroadcastReceiver





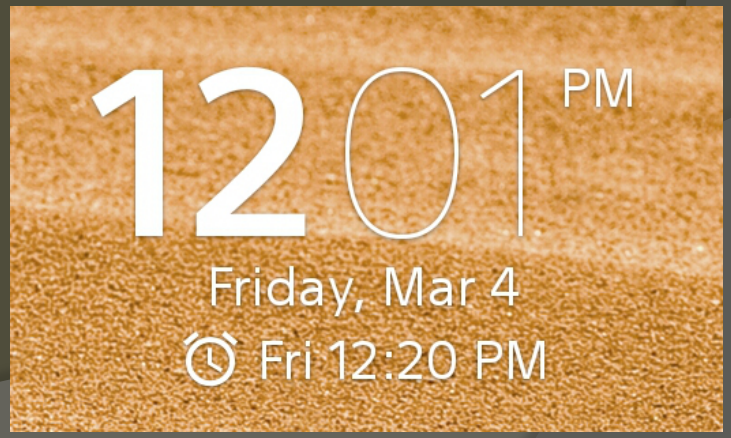


Smartphone Application

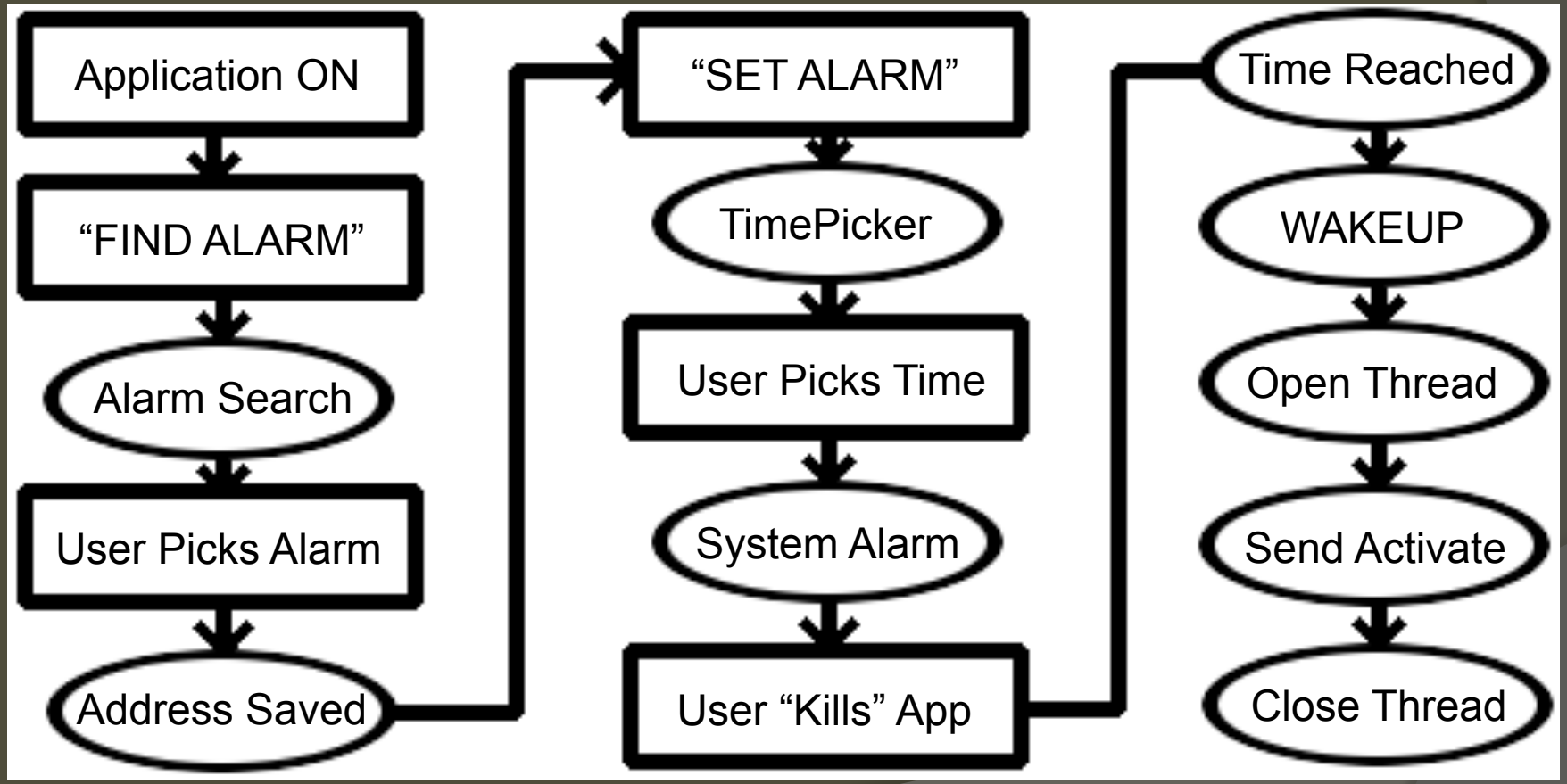
- bluetoothSearch
  - Obtains MAC Address
  - Lists nearby Bluetooth devices
  - Lets user find and pick receiver

- alarmReceive
  - Uses saved MAC Address
  - Opens Bluetooth socket
  - Sends on bit to receiver
  - Closes the connection

- Both use / implement various built-in Bluetooth Classes



Application Flow

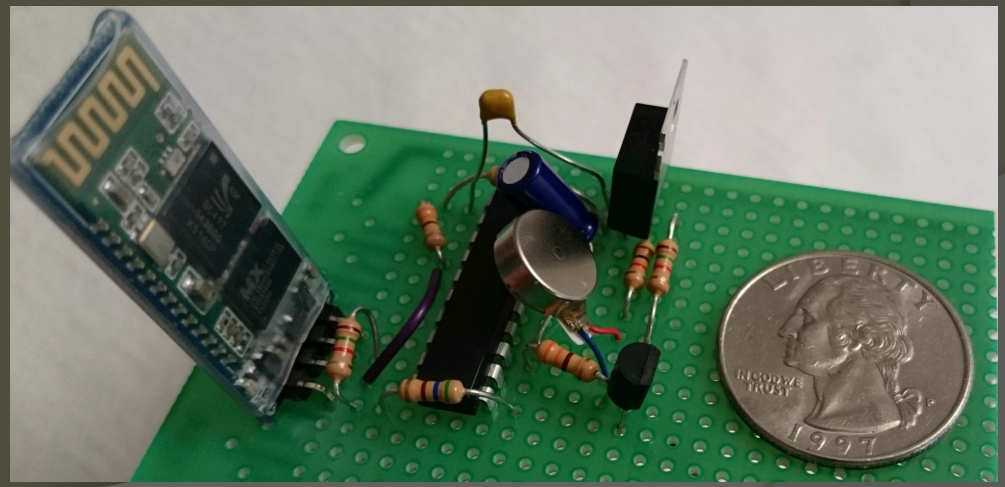


Results

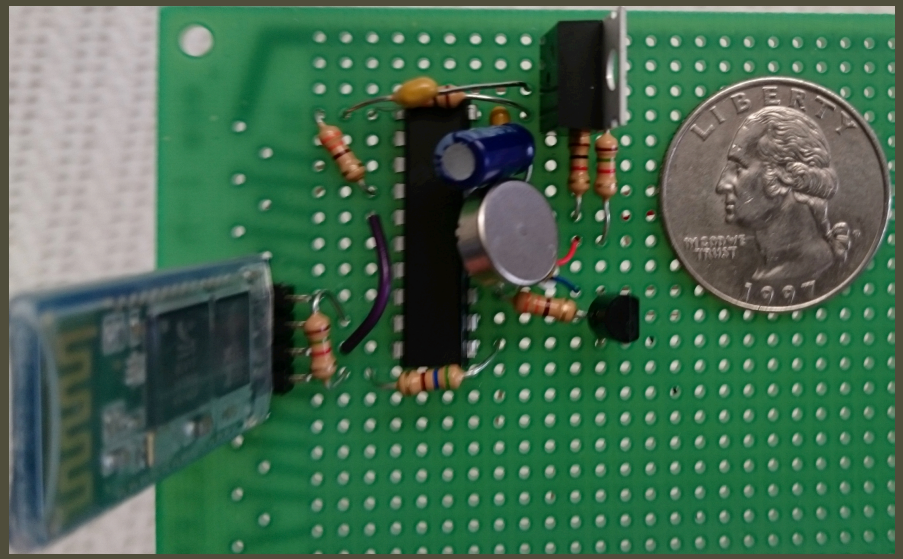
- Application (Main parts work)
  - Device Selection
  - Time Selection
  - Alarm Setting
  - Wake Up / Activate (Concern)
  - Bluetooth Enable (Concern)
  - Establish connection
  - Open Bluetooth socket
  - Send information
  - Close socket
- Arduino
  - Works, turns on motor properly
  - Simple, easy to work
- Problems:
  - Too expensive (est. >$30)
  - Too large
  - Too featured

Results

- PIC
  - Incomplete: Programming, testing, accurate current measurements
- Benefits
  - Compact, fits in 3x3cm space
  - Low current, 200mA MAX (PIC), ~3-5mA (I/O @ 5V)
  - Cost-saving (est. $15 without batteries)
  - Potentially lasts for two nights



Results



- Motor
  - Requires current gain
  - Runs at approx. 45mA
  - 3V Battery lasts 20h, good
- Bluetooth Transceiver
  - Searching: ~4-40mA
  - Connected: Drops to 3mA
  - Tx / Rx pins: 5.11 mA / .53 mA
  - Large size compared to rest
  - Inconsistent current rating

Conclusions / Future Work

- Possible, circuit and application work
- Simple for user to interface
- Software-end works
- Powering / Size issues on circuit

- Future Work
  - Find smaller versions of components
  - Smaller PIC chips
  - Build receiver into smaller circuit
  - Create circuit housing
  - Streamline application
  - Repeating alarm