goto; conference aarhus

Back to the future: sockets and relational data in your (Windows) pocket

Dragos Manolescu

~~Microsoft, Windows Phone Engineering~~

Hewlett-Packard Cloud Services

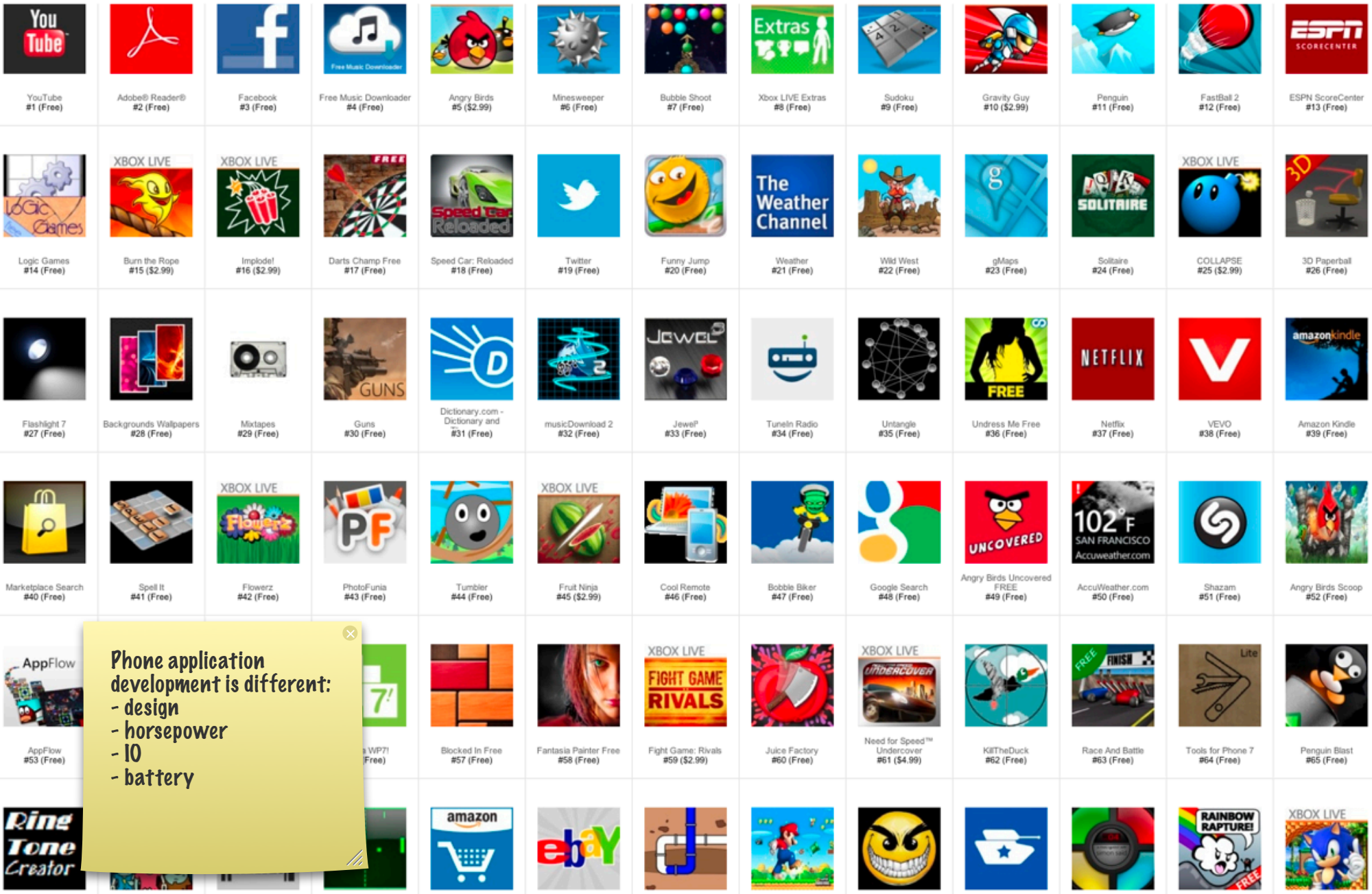
INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

gotocon.com

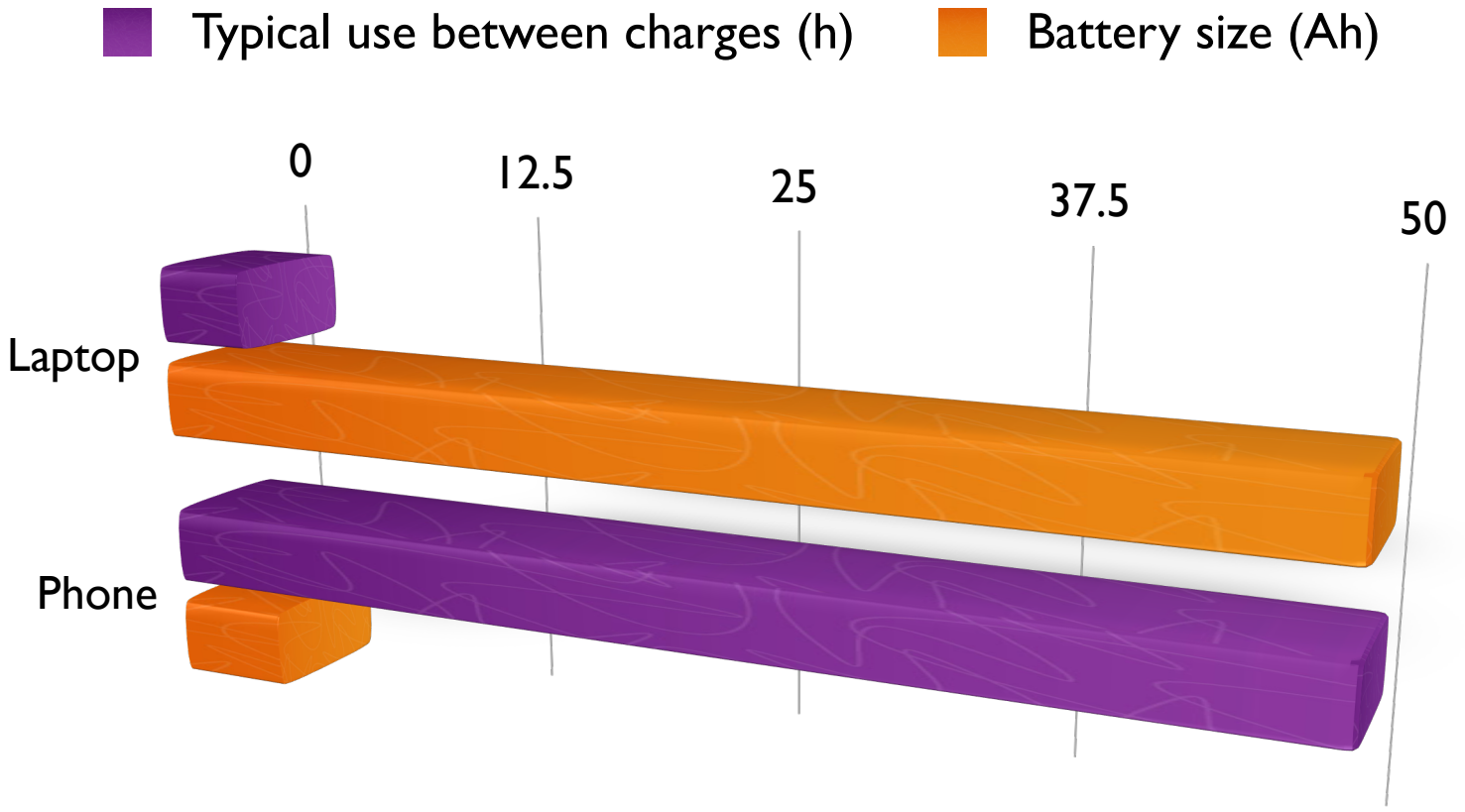
Background

APIs
Performance and Health
Data and Cloud

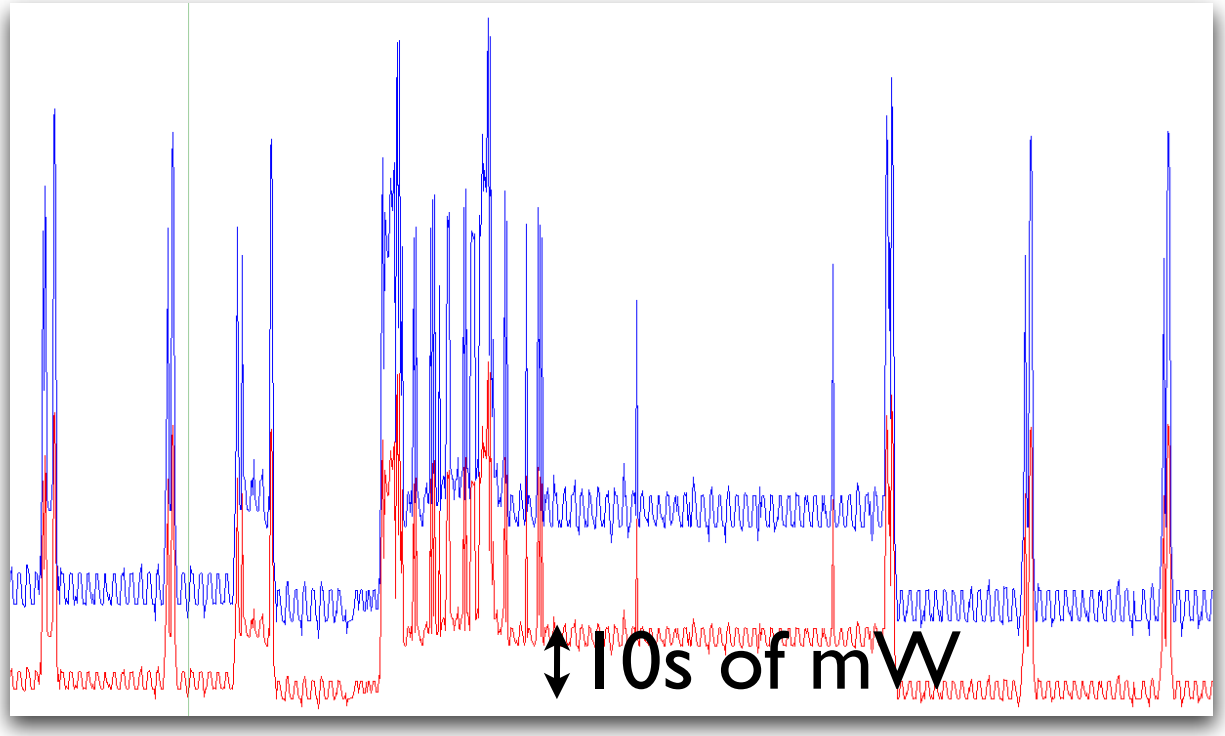

Entertainment
Shell
OS Platform
Leadership
Communications
Browser
App Platform
Partners
12:38
Phone
People
Messaging
Outlook
Pictures
XBOX LIVE
Me
DESIGN
7
Product Signoff
RTM
Networking:
WebClient and
XmlHttpRequest
Persistent data:
Isolated storage


YouTube
#1 (Free)
Adobe® Reader®
#2 (Free)
Facebook
#3 (Free)
Free Music Downloader
#4 (Free)
Angry Birds
#5 ($2.99)
Minesweeper
#6 (Free)
Bubble Shoot
#7 (Free)
Xbox LIVE Extras
#8 (Free)
Sudoku
#9 (Free)
Gravity Guy
#10 ($2.99)
Penguin
#11 (Free)
FastBall 2
#12 (Free)
ESPN ScoreCenter
#13 (Free)
Logic Games
#14 (Free)
Burn the Rope
#15 ($2.99)
Implode!
#16 ($2.99)
Darts Champ Free
#17 (Free)
Speed Car: Reloaded
#18 (Free)
Twitter
#19 (Free)
Funny Jump
#20 (Free)
Weather
#21 (Free)
Wild West
#22 (Free)
gMaps
#23 (Free)
Solitaire
#24 (Free)
COLLAPSE
#25 ($2.99)
3D Paperball
#26 (Free)
Flashlight 7
#27 (Free)
Backgrounds Wallpapers
#28 (Free)
Mixtapes
#29 (Free)
Guns
#30 (Free)
Dictionary.com - Dictionary and
#31 (Free)
musicDownload 2
#32 (Free)
Jewel²
#33 (Free)
TuneIn Radio
#34 (Free)
Untangle
#35 (Free)
Undress Me Free
#36 (Free)
Netflix
#37 (Free)
VEVO
#38 (Free)
Amazon Kindle
#39 (Free)
Marketplace Search
#40 (Free)
Spell It
#41 (Free)
Flowerz
#42 (Free)
PhotoFunia
#43 (Free)
Tumbler
#44 (Free)
Fruit Ninja
#45 ($2.99)
Cool Remote
#46 (Free)
Bobble Biker
#47 (Free)
Google Search
#48 (Free)
Angry Birds Uncovered FREE
#49 (Free)
AccuWeather.com
#50 (Free)
Shazam
#51 (Free)
Angry Birds Scoop
#52 (Free)
AppFlow
#53 (Free)
Blocked In Free
#57 (Free)
Fantasia Painter Free
#58 (Free)
Fight Game: Rivals
#59 ($2.99)
Juice Factory
#60 (Free)
Need for Speed™ Undercover
#61 ($4.99)
KillTheDuck
#62 (Free)
Race And Battle
#63 (Free)
Tools for Phone 7
#64 (Free)
Penguin Blast
#65 (Free)
Ring Tone Creator
Phone application development is different:
- design
- horsepower
- IO
- battery

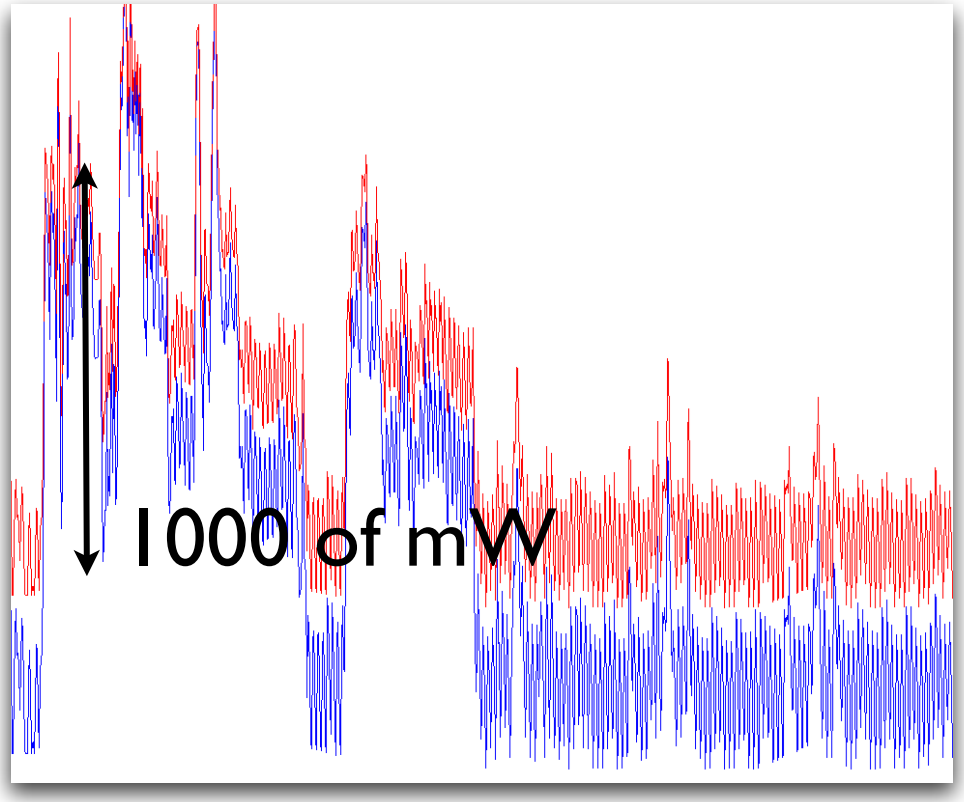
Power Usage Profiles



Touch sensor



HTTP request/response



Apple Has 1,000 Engineers Working On Chips For The Post-PC Era

ERICK SCHONFELD | posted 24 hours ago | 55 Comments

ponder what will happen to Apple **without Steve Jobs**, I keep coming back to a conversation I ew weeks ago with a veteran Silicon Valley CEO who knew Jobs. This was just after Jobs had **ed as CEO** of Apple. We got to talking about why Apple is so well-positioned in the post-PC era, and this executive zeroed in on something you don't hear too often. "Steve Jobs told me he has 1,000 engineers working on chips," he said. "Getting low power and smaller is the key to everything."

DCVS example
- CPU at 100%
- Modulate f and V
- Avoid spreading work

New in Mango (subset)

- Multitasking
- Fast application switching
- Background agents
- Sensor fusion
- Sockets
- Network information
- Device status
- Local database
- Contacts & calendar

Sockets

- Instant messaging
- Multi-player games
- Streaming

TCP

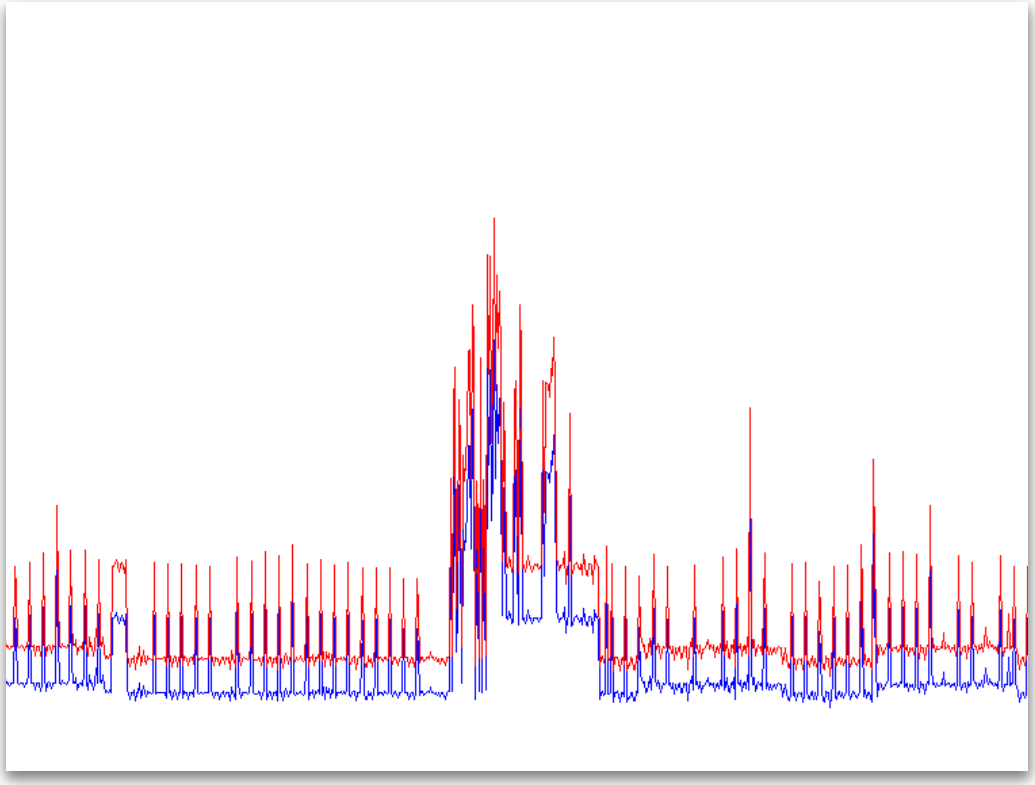
TCP Connect

```
var result = string.Empty;
var hostEntry = new DnsEndPoint("www.bing.com", 80);

_socket = new Socket( AddressFamily.InterNetwork
                    , SocketType.Stream
                    , ProtocolType.Tcp);

var socketEventArg = new SocketAsyncEventArgs
                          { RemoteEndPoint = hostEntry };
socketEventArg.Completed += (s, e) =>
                {
                    result = e.SocketError.ToString();
                };

_socket.ConnectAsync(socketEventArg);
```

Warning!

Power measurements performed on engineering devices

Current
Power

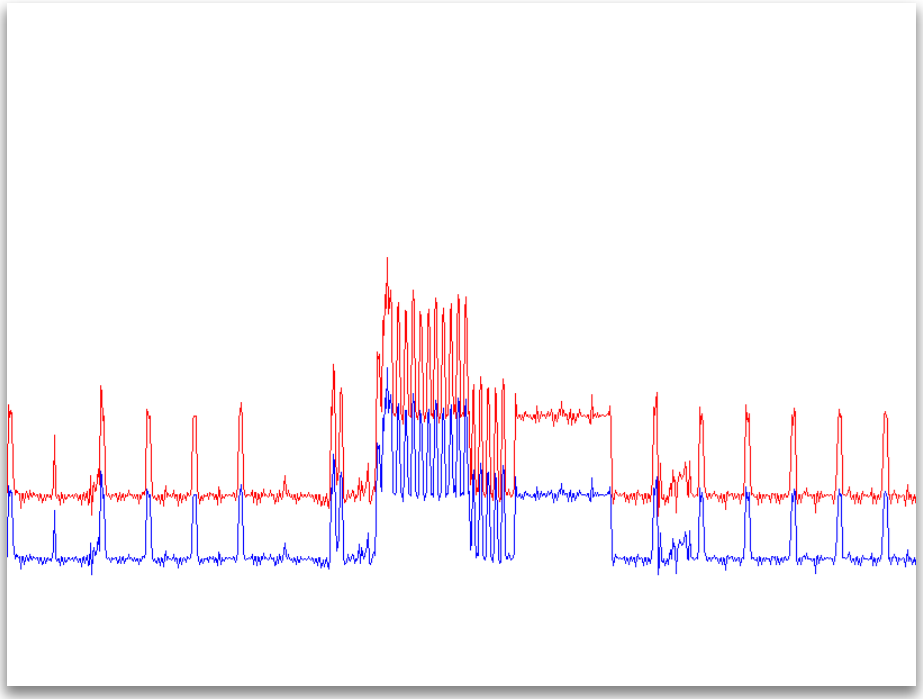


Average power: 930mW

TCP Send

```
var response = "Operation Timeout";

var socketEventArg = new SocketAsyncEventArgs
            { RemoteEndPoint = _socket.RemoteEndPoint
            , UserToken = null };

socketEventArg.Completed += (s, e) =>
                {
                    response = e.SocketError.ToString();
                    _clientDone.Set();
                };

var payload = Encoding.UTF8.GetBytes(data);

socketEventArg.SetBuffer(payload, 0, payload.Length);

_socket.SendAsync(socketEventArg);
```

Current
Power



Average power: 1400mW

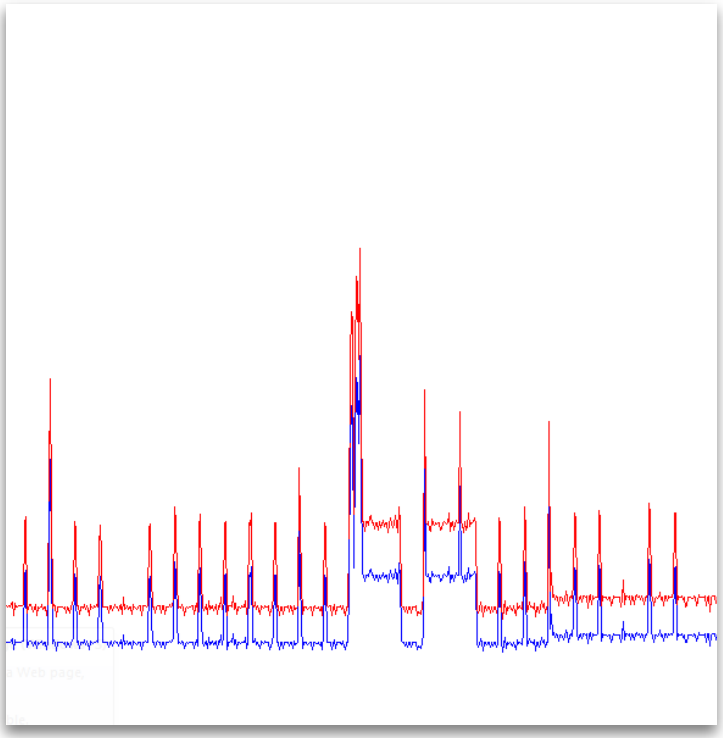
TCP Receive

```
var response = "Operation Timeout";
var socketEventArg = new SocketAsyncEventArgs
  {RemoteEndPoint = _socket.RemoteEndPoint};
socketEventArg.SetBuffer(new Byte[MAX_BUFFER_SIZE], 0,
  MAX_BUFFER_SIZE);

socketEventArg.Completed +=
  (s, e) =>
    {
      if (e.SocketError == SocketE
        response = Encoding.UTF8.
          (e.Buffer,e.Offset,e.By
      else
        response = e.SocketError
      _clientDone.Set();
    };

_clientDone.Reset();
_socket.ReceiveAsync(socketEventAr
_clientDone.WaitOne(TIMEOUT_MILLIS

return response;
```

More elegant solution with .NET 4 Task and await:

```
async Task<Byte[]> Receive()
{
// ...
return await ReceiveAsync();
}
```

Current
Power



Average power: 830mW

UDP

UDP Join Multicast Group

```
var groupAddress = IPAddress.Parse("224.0.1.1");
_client = new UdpAnySourceMulticastClient
              ( groupAddress
              , 8900);

_client.BeginJoinGroup(
                  result =>
                  {
                      _client.EndJoinGroup(result);
                      _client.MulticastLoopback = true;
                      Send("Joined the group");
                      // ready to receive
                  }, null);
```

WiFi Only

UDP Send/Receive

```
// Send
_client.BeginSendToGroup(data, 0, data.Length,
    result => _client.EndSendToGroup(result), null);

// Receive
_client.BeginReceiveFromGroup(recBuffer, 0, recBuffer.Length,
    result =>
          {
           _client.EndReceiveFromGroup(result, out source);
           string dataReceived = Encoding.UTF8.GetString
               (_receiveBuffer, 0, _receiveBuffer.Length);
           Receive(); // next message from the group
          },
    null);
```

WiFi Only

Network Preferences

Connect to specific interface

```
sock = new Socket( AddressFamily.InterNetwork
                 , SocketType.Stream
                 , ProtocolType.Tcp);
var netList = new NetworkInterfaceList();
foreach (NetworkInterfaceInfo info in netList)
{
  if (info.InterfaceType == NetworkInterfaceType.Wireless80211)
        {
          sock.AssociateToNetworkInterface(info); // extension
          break;
        }
}
```

Query interface

```
private void SocketEventArg_Completed(object sender,
  SocketAsyncEventArgs args)
{
 if (args.LastOperation == SocketAsyncOperation.Connect &&
     args.SocketError == SocketError.Success)
    {
     var ifName = sock.GetCurrentNetworkInterface() // extension
                       .InterfaceName;
    }
}

// similar for WebRequest
```

Network change notification

```
DeviceNetworkInformation.NetworkAvailabilityChanged +=
  (src, networkArgs) =>
{
     /* name, bandwidth, characteristics */
 NetworkInterfaceInfo nInterface = networkArgs.NetworkInterface;

     /* characteristic update (roaming, type, bandwidth),
connected, disconnected */
 NetworkNotificationType nType = networkArgs.NotificationType;
};
```

Name resolution

```
var endpoint = new DnsEndPoint("www.bing.com", 80,
   AddressFamily.InterNetwork);
var nrResult = new NameResolutionResult();
var ninterface = default(NetworkInterfaceInfo);
DeviceNetworkInformation.ResolveHostNameAsync(endpoint,
   ninterface, /* resolve on this interface */
   (NameResolutionResult result) =>
                        {
                            nrResult = result;
                            _waitEventHandle.Set();
                        }, null);

_waitEventHandle.WaitOne();

var ipEndPoints =
  from ipEndPoint in nrResult.IPEndPoints
  where ipEndPoint.AddressFamily == AddressFamily.InterNetwork
  select ipEndPoint;
```

Net preference/requirement

```
SetNetworkPreference(this Socket
socket, NetworkSelectionCharacteristics preference)
SetNetworkRequirement(this Socket
socket, NetworkSelectionCharacteristics requirement)
NetworkInterfaceInfo GetCurrentNetworkInterface(this Socket
socket)

SetNetworkPreference(this WebRequest
request, NetworkSelectionCharacteristics preference)
SetNetworkRequirement(this WebRequest
request, NetworkSelectionCharacteristics requirement)
NetworkInterfaceInfo GetCurrentNetworkInterface(this WebRequest
request)

NetworkSelectionCharacteristics: Cellular, NonCellular
```

Developer Notes

- Old and new code (UDP SendTo, ReceiveFrom)
- Two async models (completed event, IAsyncResult)
- Phone integration points:
  - Connection manager
  - Fast application switching
  - Background services
  - Network preferences: interface state, type, subtype, characteristics

Local Storage

Database

- Offline data cache
- Reference data
- Search/sort quickly
- Application state
- Schema changes for application updates

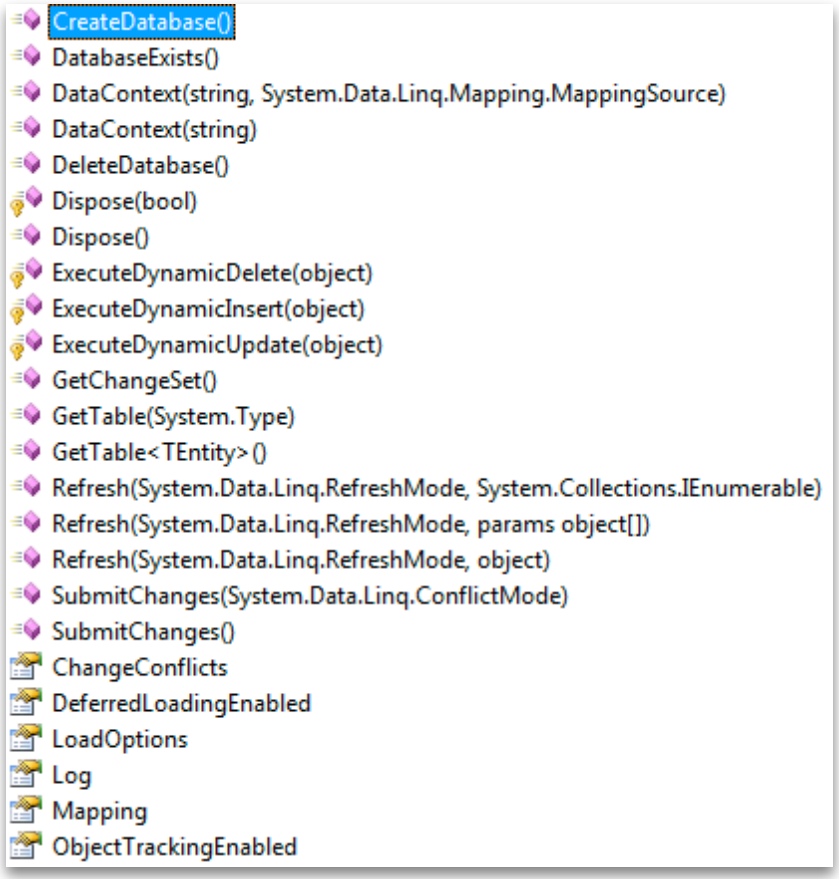
LINQ-to-SQL API

- DataContext: entry point for
  - Database management
  - Composing queries
  - Issuing changes
- Mapping Attributes:
  - Classes and members <-> tables and columns
  - MetaModel for change processor
- ITable, ITable<T>: IQueryable and IQueryProvider

Data Context

```
public class ToDoDataContext : DataContext
{
 public ToDoDataContext(string connStr)
             : base(connStr)
         { }

 public Table<ToDoItem> Items;
 public Table<ToDoCategory> Categories;
}


using (ToDoDataContext db =
 new ToDoDataContext("isostore:/ToDo.sdf"))
{
    if (db.DatabaseExists() == false)
    {
        db.CreateDatabase();
    }
}
```

Code-first



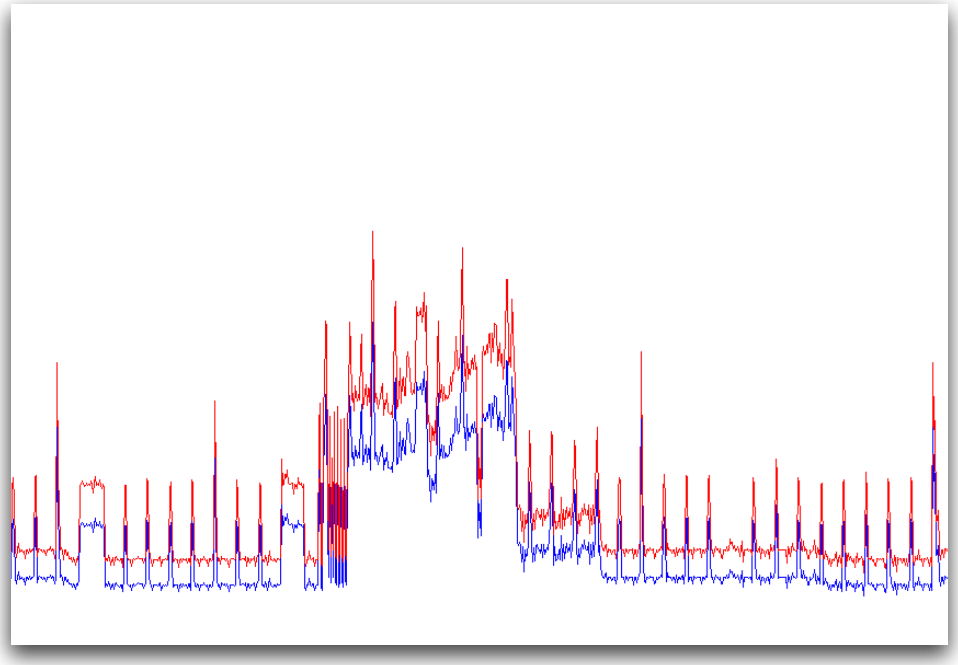
Mapping

```
[Table]
public class ToDoItem : INotifyPropertyChanged, INotifyPropertyChanging
{
  [Column(IsPrimaryKey = true, IsDbGenerated = true, DbType = "INT NOT
NULL Identity", CanBeNull = false, AutoSync = AutoSync.OnInsert)]
  public int ToDoItemId { /* … */ }

  [Column]
  public string ItemName { /* … */ }

  [Column]
  public bool IsComplete { /* … */ }

  // Version column aids update performance.
  [Column(IsVersion = true)]
  private Binary _version;

  // Association between this key and that "storage" table
  [Association(Storage = "_category", ThisKey = "_categoryId", OtherKey =
"Id", IsForeignKey = true)]
  public ToDoCategory Category { /* … */ }
    }
```

Insert

```
var newToDoItem = new ToDoItem { /* ... */ };

toDoDB.Items.InsertOnSubmit(newToDoItem);
toDoDB.SubmitChanges();
```

Current
Power

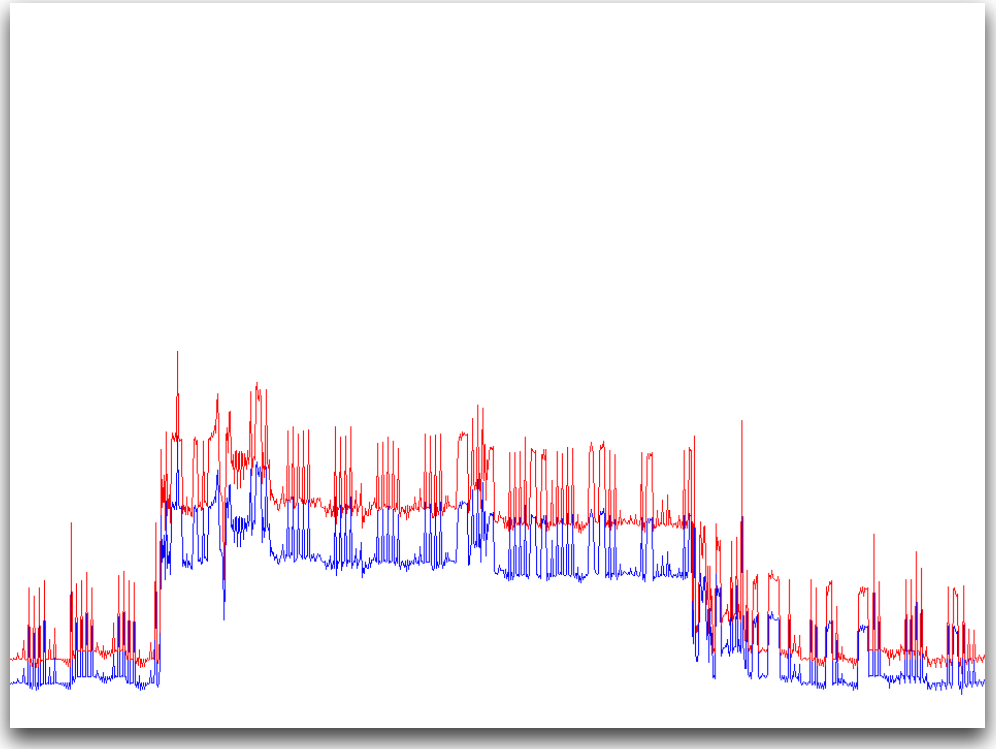


Average Power ≈ 500mW

Query

```
var toDos = from ToDoItem todo in toDoDB.Items
            where todo.IsComplete
            select todo;
```

Current
Power

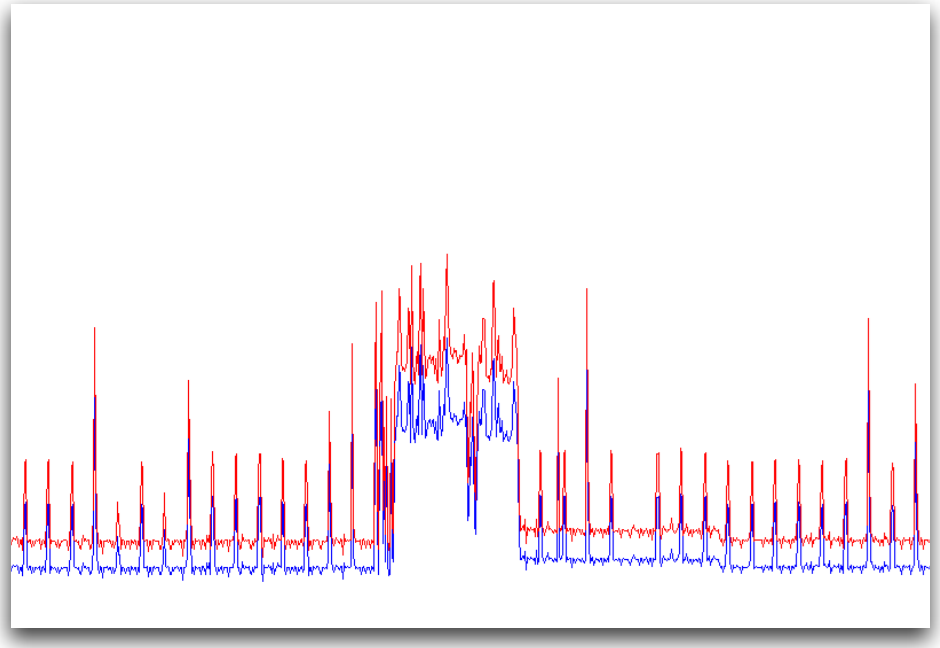


Average Power ≈ 500mW

Update

```
var updateItem = (from ToDoItem todo in toDoDB.Items
                  where todo.ToDoItemId == 42
                  select todo)
                 .First();

updateItem.IsComplete = false;
toDoDB.SubmitChanges();
```

Current
Power

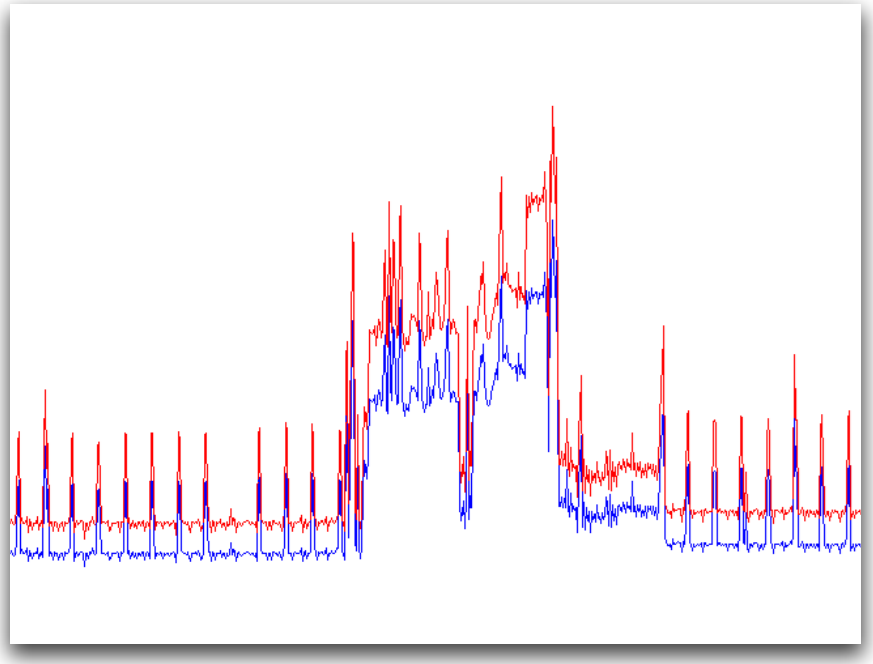


Average Power ≈ 500mW

Delete

```
toDoDB.Items.DeleteOnSubmit(toDoForDelete);
toDoDB.SubmitChanges();
```

Current
Power



Average Power ≈ 500mW

Other Local Data

Contacts data sources

Provider	Name	Picture	Other	Appts
WP	✔	✔	✔	✔
WL Social	✔	✔	✔	✔
WL Rolodex	✔	✔	✔	✔
Exchange	✔	✔	✔	✔
MO	✔	✔	✔	✘
facebook	✔	✔	✘	✘
WL Agg	✘	✘	✘	✘

WL = Windows Live

Contacts

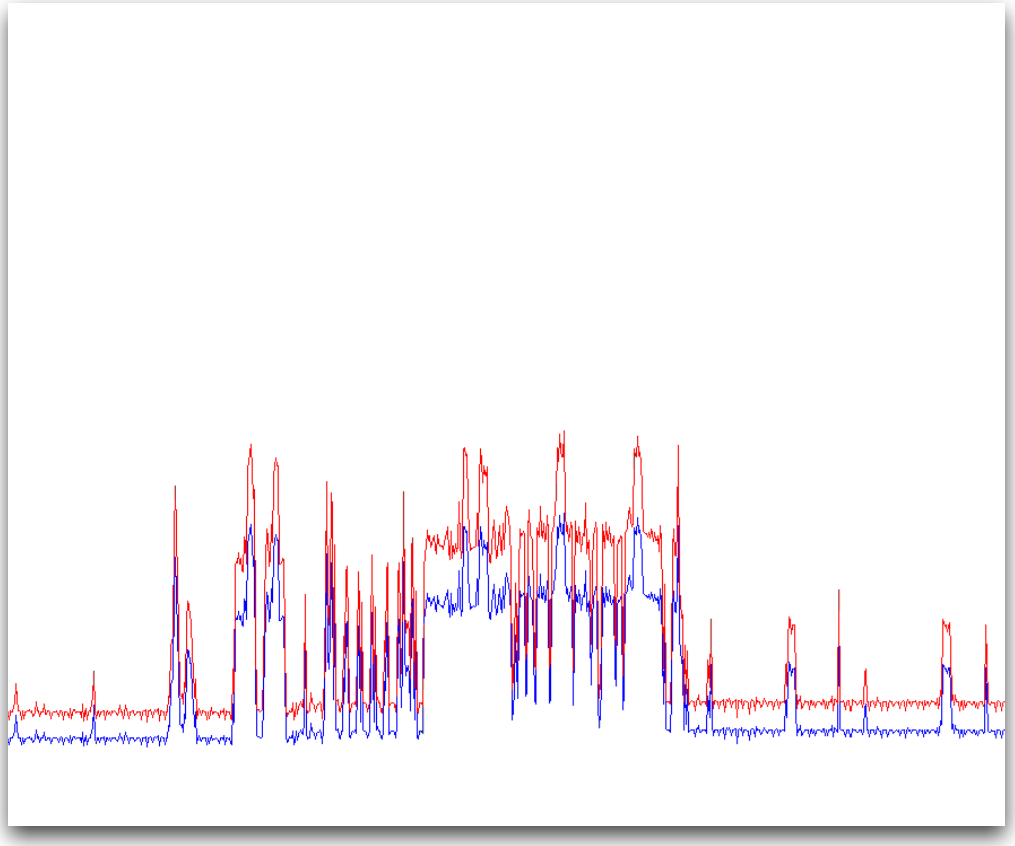
Read-only access

Built-in filters are pre-indexed

LINQ possible; bypass filters

```
var cons = new Contacts();

cons.SearchCompleted += (s, scea) =>
    {
      ContactResultsData.DataContext = scea.Results;
      var textBlock = (TextBlock) scea.State;
      ContactResultsLabel.Text =
                      ContactResultsData.Items.Count > 0
                       ? "results (tap name for details...)"
                       : "no results";
    };

cons.SearchAsync( "google"
                , FilterKind.EmailAddress
                , ContactResultsLabel
                );
```

Current
Power

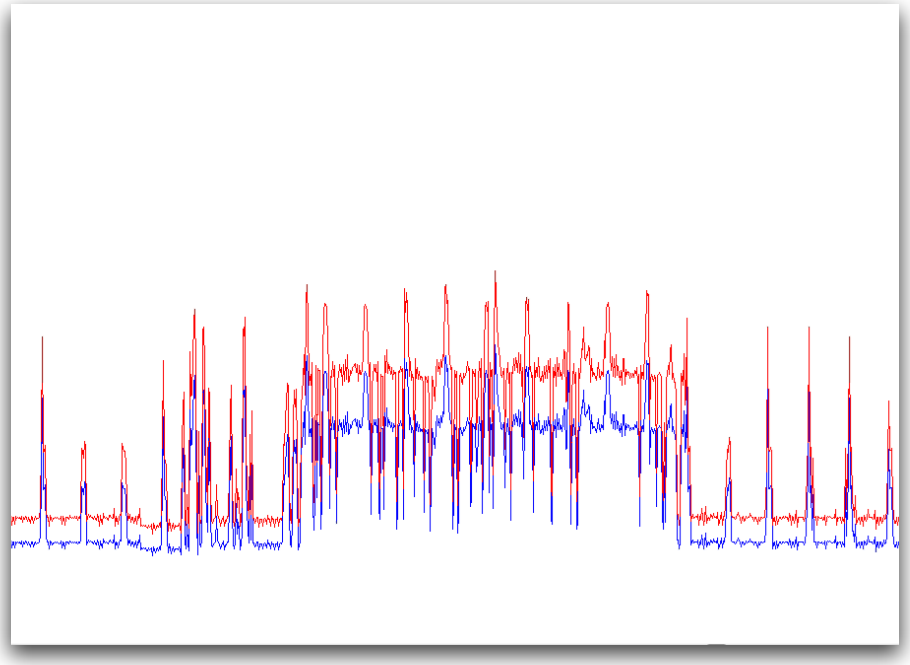


Average Power ≈ 500mW

Contacts with LINQ

```
Contacts cons = new Contacts();

cons.SearchCompleted += (s, e) =>
{
  ContactResultsDataLINQ.DataContext =
            from Contact con in e.Results
            from ContactEmailAddress a in con.EmailAddresses
            where a.EmailAddress.Contains("google")
            select con;
};

cons.SearchAsync(String.Empty, FilterKind.None, null);
```

Current
Power

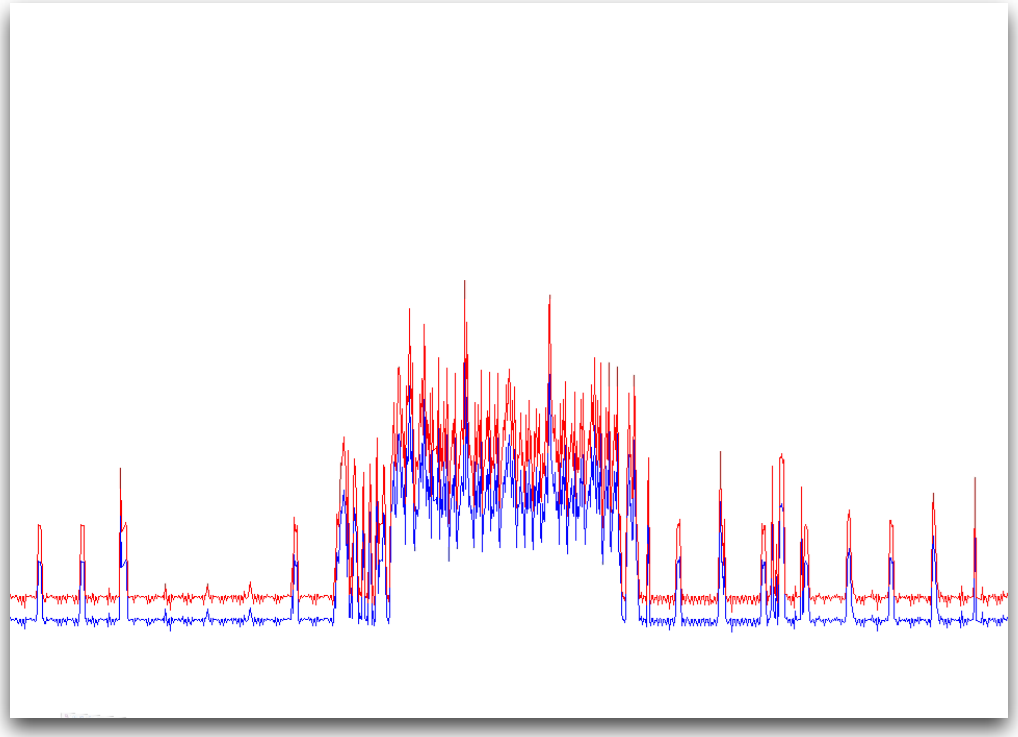


Average Power ≈ 500mW

Appointments

```
var appts = new Appointments();

appts.SearchCompleted +=
 (s, asea) =>
  {
    StartDate.Text =
asea.StartTimeInclusive.ToShortDateString();
    EndDate.Text = asea.EndTimeInclusive.ToShortDateString();
    AppointmentResultsData.DataContext = asea.Results;
    var textBlock = (TextBlock) asea.State;
    textBlock.Text = AppointmentResultsData.Items.Count > 0
                       ? "results (tap for details...)"
                       : "no results";
  };

 var start = DateTime.Now;
 var end = start.AddDays(7);
 appts.SearchAsync(start, end, 20, AppointmentResultsLabel);
```

Current
Power



Average Power: ≈500mW

Developer Notes

- ExecuteCommand: not supported
- ADO.NET Objects (such as DataReader): not supported
- Only SQL CE data types are supported
- Data binding:
  - Table.IListSource.GetList Method is not supported
  - BinaryFormatter is not supported
- Take() requires a constant argument in LINQ queries
- Skip() and Take() require an ordered list
- Built-in searches vs LINQ

Summary

- New features:
  - TCP and UDP sockets
  - Network Preferences
  - Local storage
- Power profiles
- Idiosyncrasies and developer notes

Next Steps

- SDK download: http://j.mp/paVn3s
- Mango new features: http://j.mp/nrMGRj
- .NET4 Task<T>: http://j.mp/pf5yJq
- Local db best practices: http://j.mp/qviI4F

Thank you!

@hysteresis