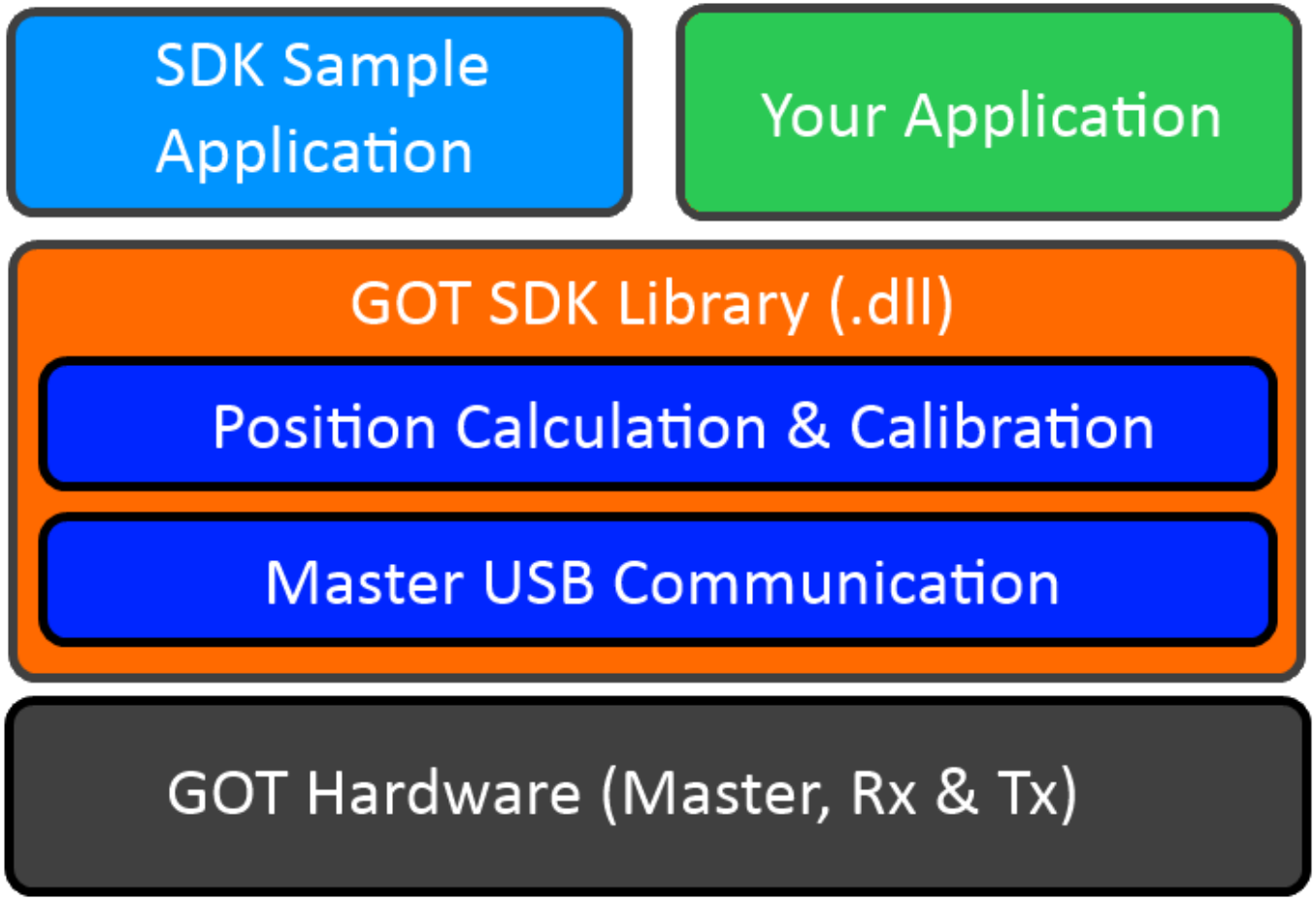
GamesOnTrack SDK

Presentation & Overview

The purpose of the SDK

- Allow other developers to create applications using the GOT position system
- Flexible component-based system
- The SDK is just a .NET class library. Build whatever you want on top of it.
- Used in our own product GTCommand / Faller Digital Car System

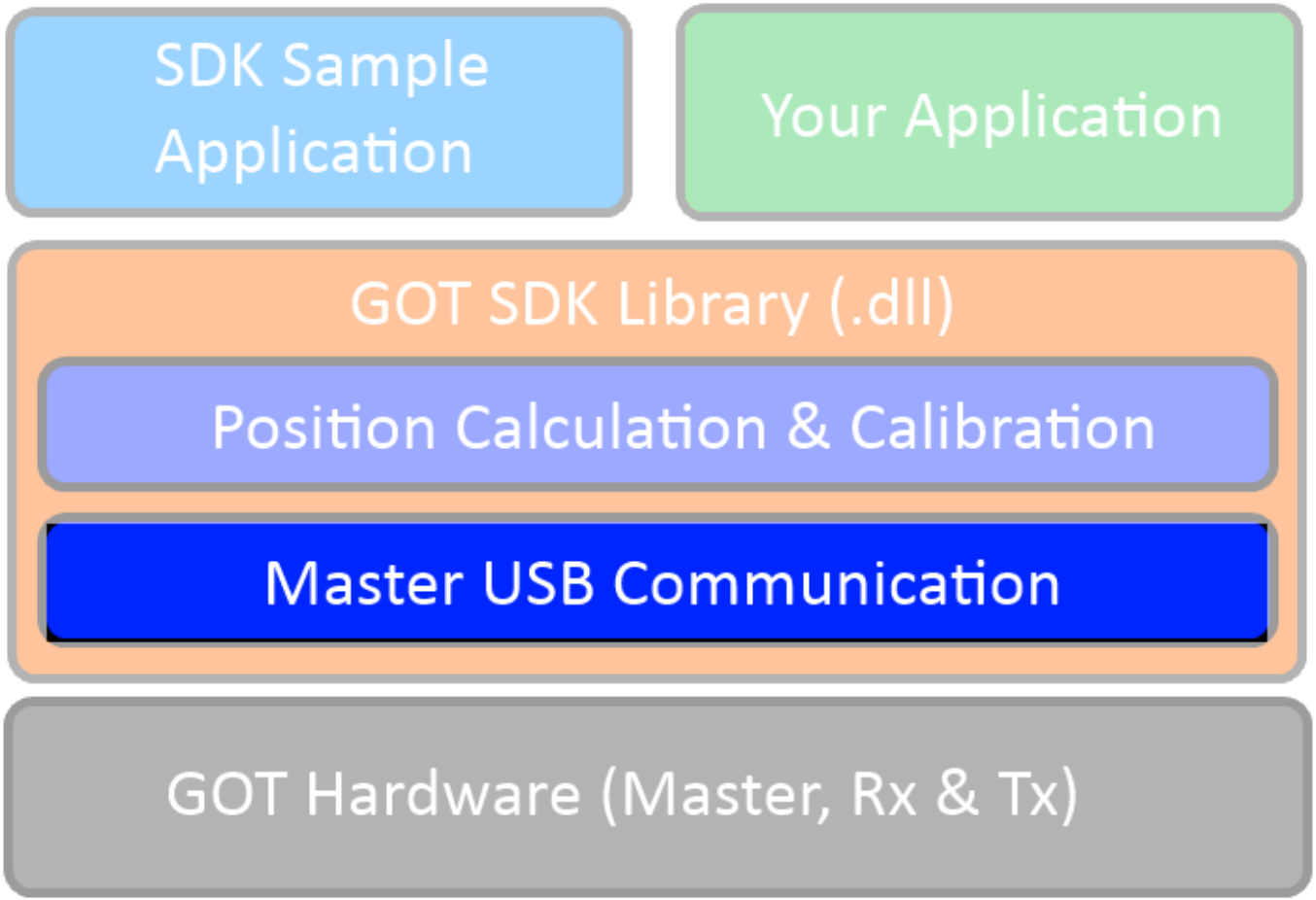
Architecture



SDK Requirements

- .NET 4.0 Framework (Full, not Client Profile)
- Third Party USB Driver (Silabs)
- Sample Application: Visual Studio 2010 or later (Express is okay)
- Mono currently unsupported (due to the use of System.Management and System.Windows.Media). Linux/Mac Silabs USB driver does exist.

SDK Master USB Communication



Responsibilities

- Detect USB-port, open connection, etc.
- Low-level communication
- Receive data (new connected units, measurement data)
- Send data (setup configuration, temperature)

Basic Concepts

- Transmitter, Receiver, Measurement
- Unique id on everything (”GOT Address”)
- Event-based API
  - OnMasterStatusChanged(Status s)
  - OnTransmittedConnected(Transmitter t)
  - OnReceiverConnected(Receiver r)
  - OnMeasurementReceived(Measurement m)

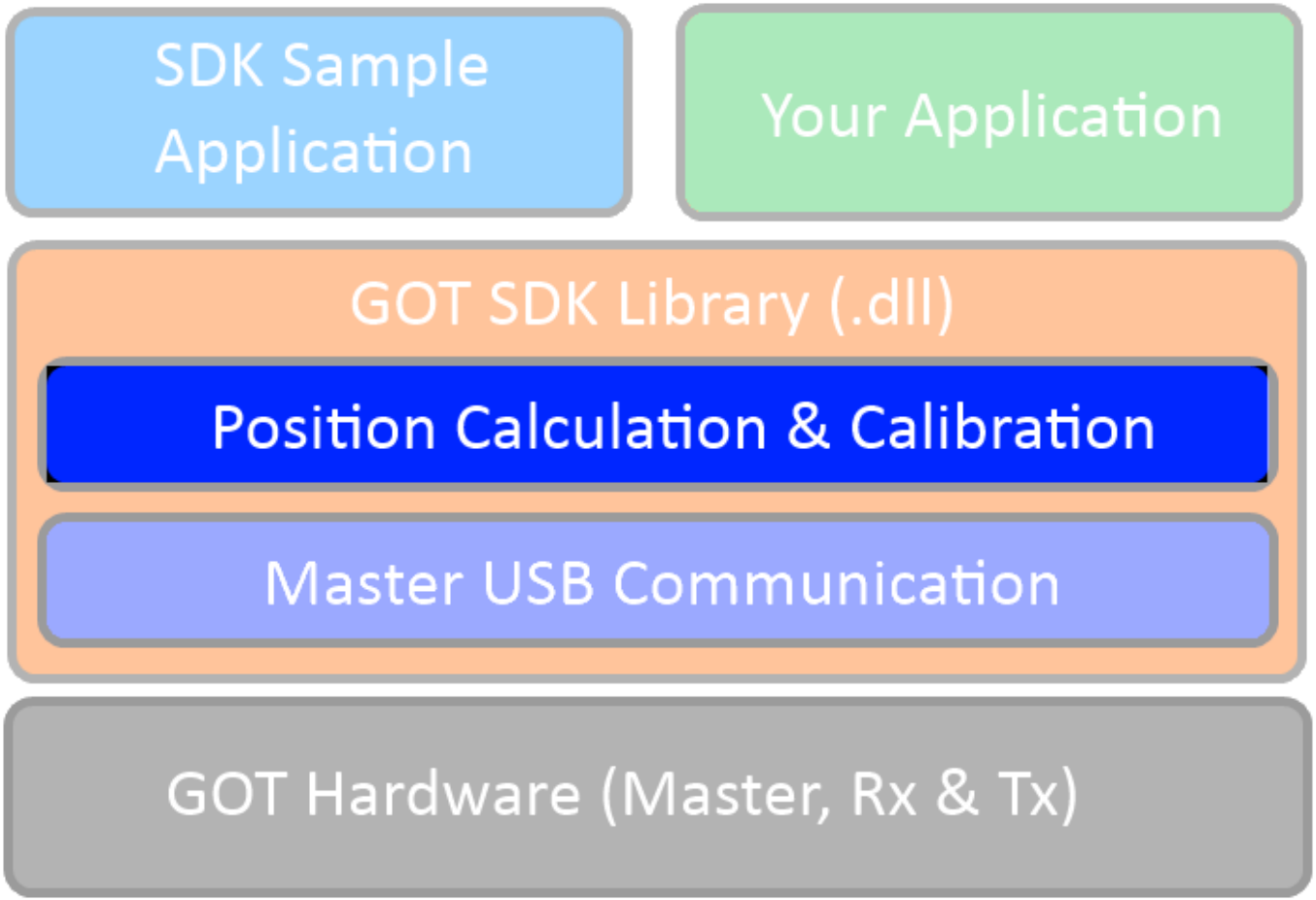
Measurement Data

```
// Slightly simplified version of the class (e.g.  Battery  info removed)
class Measurement
{
	public  GOTAddress TxAddress { get; set; }
	public  byte RSSI { get; set; }
	public  byte RadioQuality { get; set; }  // [0..100%]

	// Data for the individual measurements to each receiver.
	public ReadOnlyCollection<Rx>  RxMeasurements { get; set; }

	struct Rx
	{
		public  GOTAddress RxAddress;
		public  int RSSI;
		public  int Level; // 0..255
		public  int DistanceMM;
	}
}
```

SDK Position Calculation



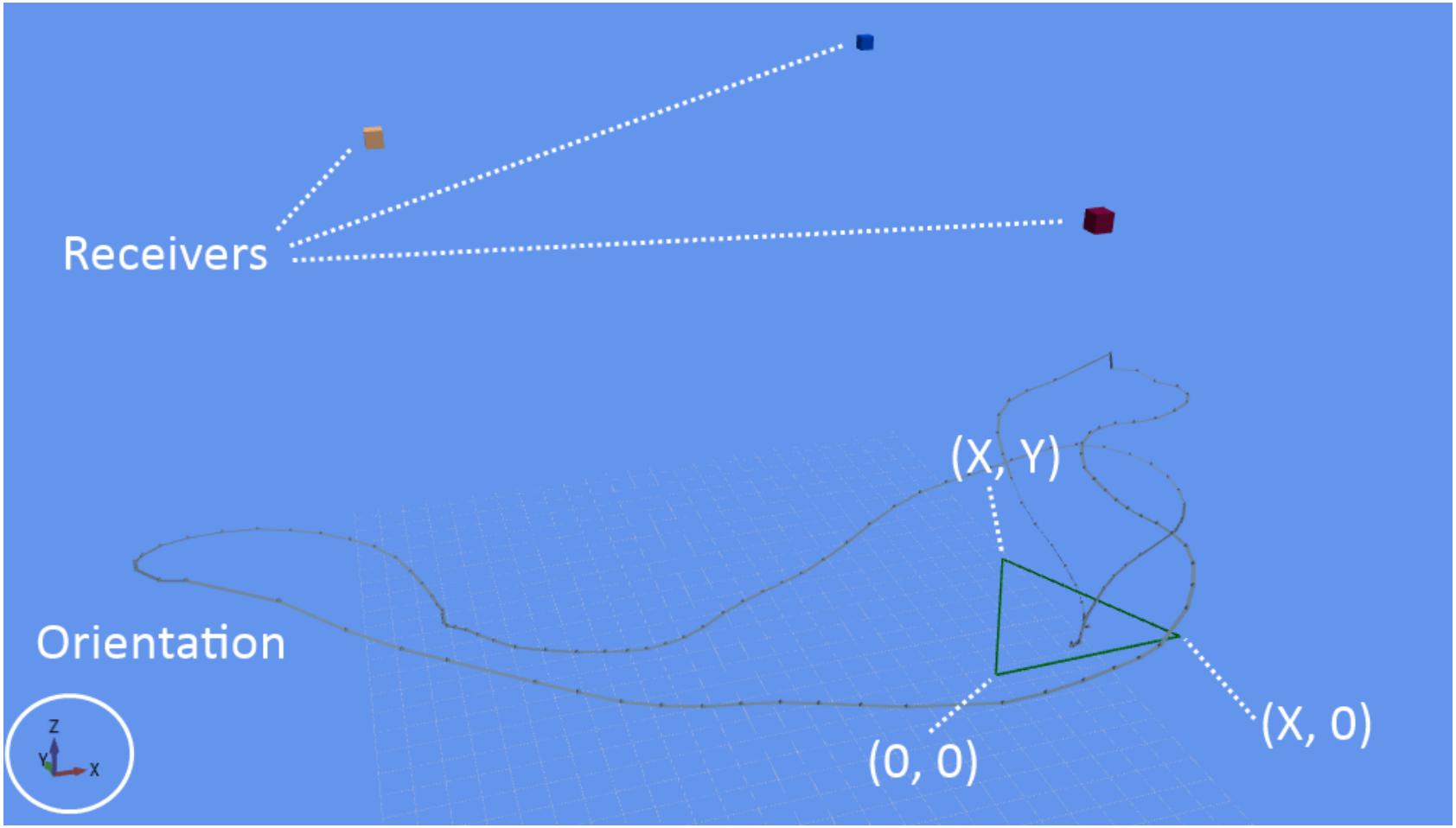
Responsibilities

- This part of the SDK is optional
- Converting the measured distances into (X, Y, Z) coordinates
- The concept of ”Trilateration”
- Two phases
  - Calibration: Create the coordinate system. Only needs to be done once.
  - Calculation: Use the calibration in conjuction with a measurement to find (X, Y, Z)

The ”Scenario” Concept

- Represents a (calibrated) coordinate system
- The SDK uses the ”auto calibrator”, but it can be done manually as well
- 3 transmitter positions + 3 receivers => 9 distances in total
- See illustration on next slide

Calibration Overview



Detecting the Auto Calibrator

```
private CalibratorTriangle calibrator = null;
private List<Transmitter>  connectedTransmitters = new List<Transmitter>();
// Called when a new transmitter has been detected
private void master_OnNewTransmitterConnected(Transmitter newTransmitter)
{
    this.connectedTransmitters.Add(newTransmitter);

    // Search for auto calibrator (uses a hardcoded range of ids)
    bool found =
    CalibratorTriangle.TryFindCalibratorTriangle(this.connectedTransmitters, out
    this.calibrator);

    if (found)
    {
        // Ready to start calibration
    }
}
```

Calibration

```
// Called when a new measurement is received (usually every 100 ms)
private void master_OnMeasurementReceived(Measurement measurement)
{
    this.calibrator.AddMeasurement(measurement);

    double progress; // Status progress [0...1]
    if (this.calibrator.IsCalibrationFinished(out progress))
    {
        // TODO: Save the calibrated scenario somewhere
        Scenario3D calibratedScenario = calibrator.CreateScenario();
    }
    else
    {
        // TODO: Display progress somewhere in UI
    }
}
```

Using a Scenario

```
// Called when a new measurement is received (usually every 100 ms)
private void master_OnMeasurementReceived(Measurement measurement)
{
    Scenario3D myCalibratedScenario = GetScenarioSomehow(...);

    CalculatedPosition cp;
    if (PositionCalculator.TryCalculatePosition(measurement,
    myCalibratedScenario, out cp))
    {
        // TODO: Do something with pos
        Point3D pos = cp.Position;
    }
}
```

Extending a Scenario

- Adding additional receiver
  - Measure distance to a transmitter in multiple locations. Find position of a new receiver.
- Merging multiple scenarios
  - Promote one to ”main” scenario
  - Find transform matrix from other scenarios to main and offset into a global coordinate system.

SDK Sample Application