LIGHT CATCHER DEMO CONTROL

NI LABVIEW DEVELOPER DAYS 2018

STIJN HELSEN

04/06/2018

FLANDERS MAKE

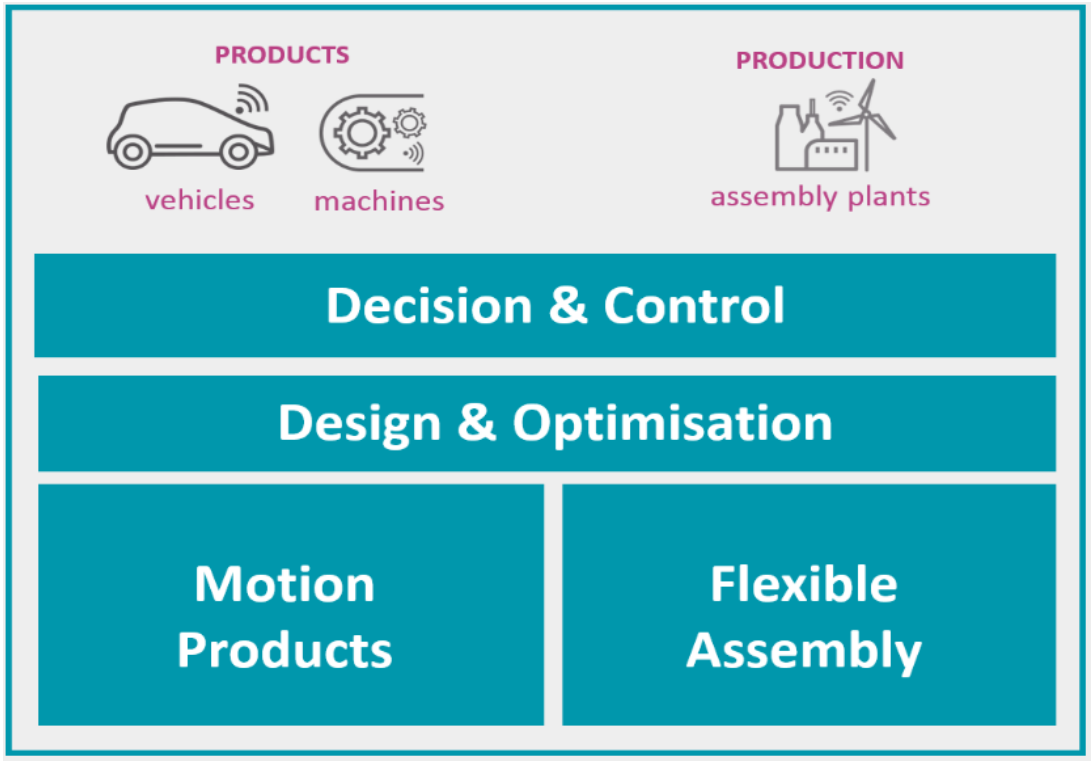
MANUFACTURING INNOVATION NETWORK

Contents

- Flanders Make
  - Mechatronics 4.0
- Context
- Algorithm Development
- Testing
- Demo

Flanders Make

- Strategic Research Centre in Flanders for "Make Industry"
  - (Leuven – Lommel - …)
- All based on modelling & virtualisation
- Collective projects
  B2B projects



As a growing organisation – we always have job offers!
see: http://www.flandersmake.be/nl/jobs

Position of this project "Mechatronica 4.0"

- Support Flemish SME's (active in the make industry) in the integration of "Industry 4.0" (the fourth industrial revolution)
- Cooperation between
  - Sirris
  - Flanders Make
  - iMinds (fused with imec)
- Small "sub-projects" with companies
  - Using knowledge/experience for faster results
  - Part of the project results (the generic results) may be used elsewhere

Context

- Light catcher
  - Rotating mirror on a roof
    - increase the amount of light reflecting sunlight downward (especially when the sun is low above the horizon)
    - standalone (no external power supply)
- Development of “light catcher” control
  - Robust against variable weather conditions
  - Easy installation (free orientation, no complex calibration procedure, …)

EcoNation®

Light catcher – related components

- What's modelled/simulated?
  - Light sensor ("sun position sensing")
  - Position sensor (pot-meter)
  - Rotation (DC-motor)
  - Solar cell (only for estimating electrical input power, not used in control)

Algorithm Development

- Target:
  - Develop control, reuse code in an (existing) embedded controller
    ➔ C-code needed

    Proto/testing → project – avoid recoding (as much as possible)
    ➔ No LabVIEW
- Solution:
  - ➔ Algorithm developed in Matlab
    → autocoding to C
  - ➔ Simulation of sun movement
    position on earth, season, weather dependencies
  - ➔ Simulation of sensors and rotation
    (for characteristics of sensors, LabVIEW was used)
    → "LabVIEW was caring for the Hardware part"

Hardware I/O

- Inputs
  - Light sensor: 2 voltages from photo diodes (“light sensor”)
    + 1 voltage from pot-meter (rotation)
- Outputs
  - Motor: left/right rotation / standstill (DC-motor)

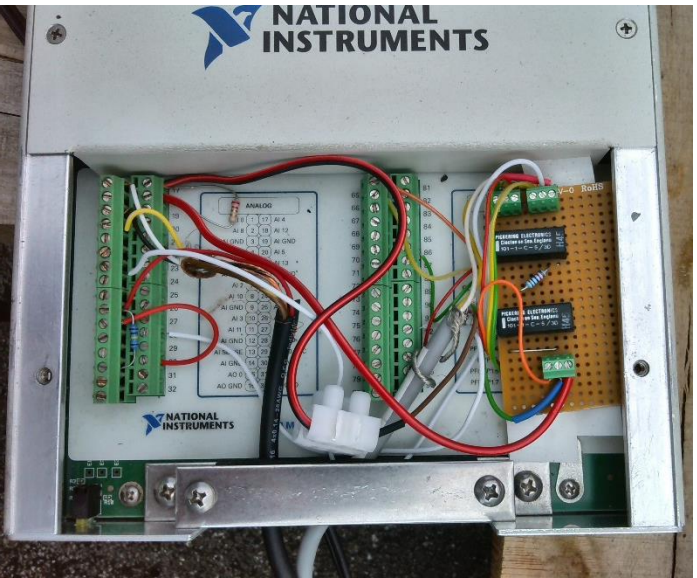
- Additional input
  - Voltage from solar cell (to estimate power supply to charge battery)

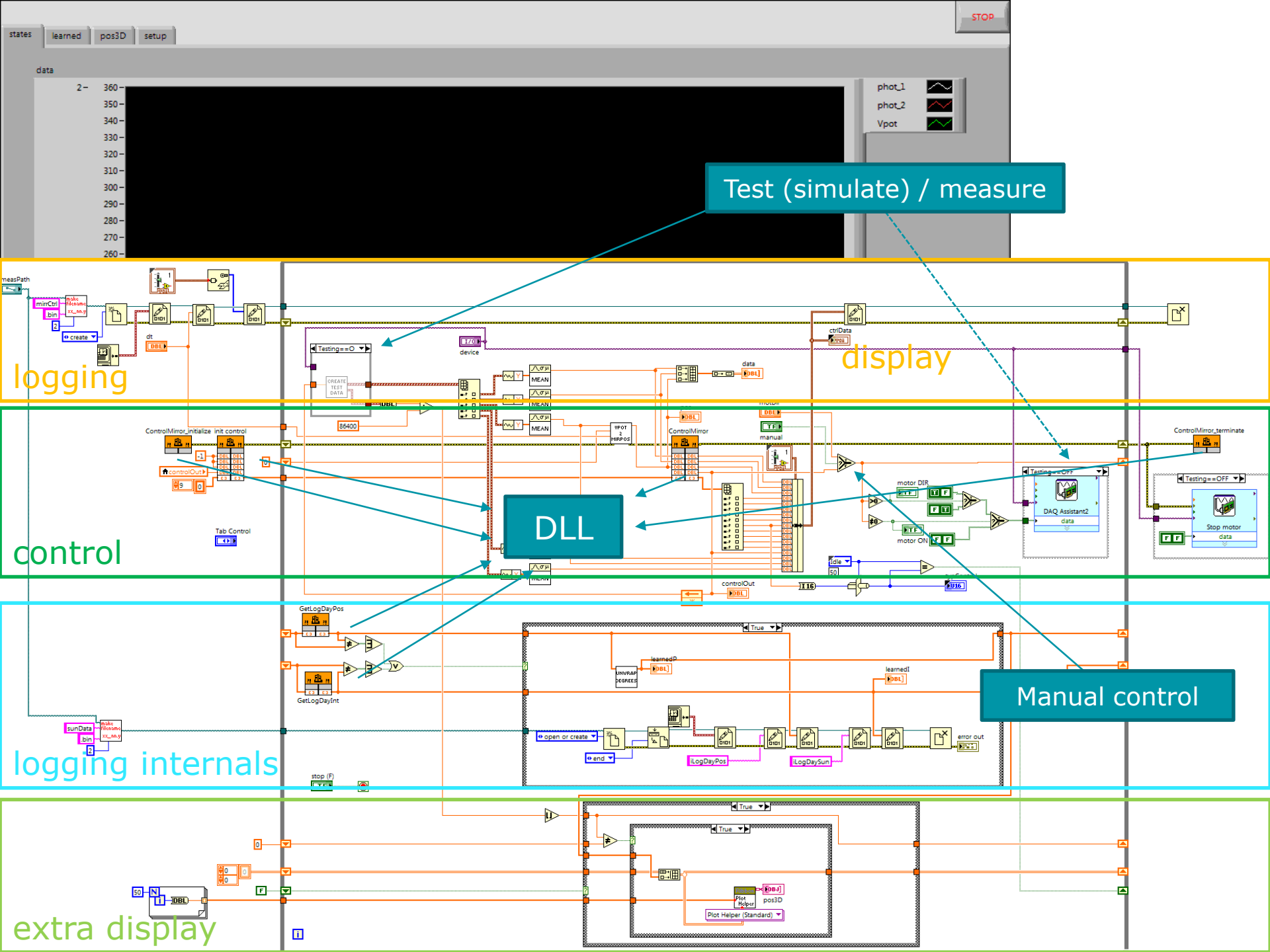
Testing → real world

- Requirements for test system
  - Able to use Matlab- or C-code
  - Flexible with hardware
    - Able to read analogue signals (high impedance and accurate enough)
    - Able to control a motor (with not too much hardware)
  - Preferably available at Flanders Make
  - Easy to use, flexible and able to log easily
- Choice
  - NI USB-6251 (generic DAQ system)
    - With some reed relays to switch motor (using 5V from USB-6251)
  - C-code compiled to a DLL, used within LabVIEW

Test setup

- Light catcher with USB-6251 on the roof (covered)
- Power (220V) and USB-cable to the inside
- PC inside with control software
  - Control & logging
  - Separate VI copying data regularly to the network
    - ➔ robust logging (internally)
    - ➔ easy to follow up via other computer

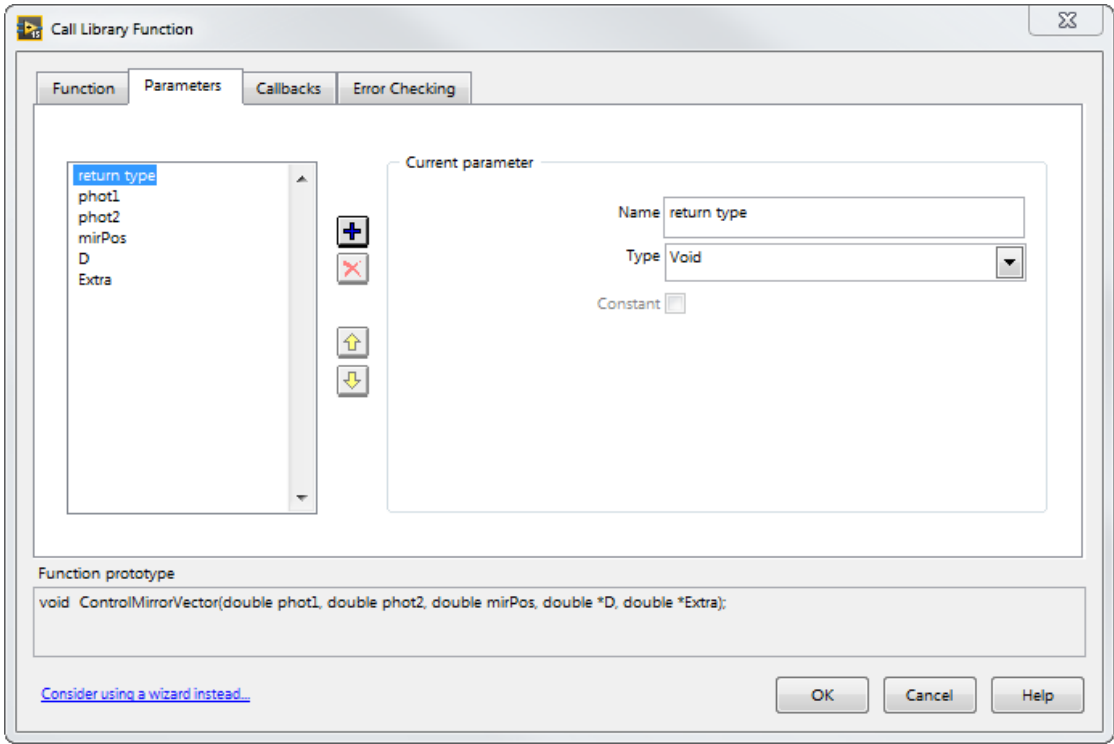



states
learned
pos3D
setup
STOP
data
phot_1
phot_2
Vpot
Test (simulate) / measure
logging
display
control
DLL
logging internals
Manual control
extra display

Control algorithm

- Matlab – call syntax:
  - `function [T,Extra]=ControlMirrorVector(phot1,phot2,mirPos)`

    T: -1, 0, +1 (motor rotating / stationary)
- C – call syntax:
  - `void ControlMirrorVector(real_T phot1, real_T phot2, real_T mirPos, real_T *D, real_T Extra[10])`
- LabVIEW



And it worked….

- “Can you give a demo on the ‘Flanders Make Symposium’?”

and soon!

and without too much effort…

Demo

- Difficulties
  - Show learning control over days in minutes?!
  - Show “following the sun” inside?!
- Solution
  - Create a “fast moving sun”
    - Rotating light (one plane)
    - Intensity control
  - Acceleration factor (in “sun control” but also in product code): 100x
    - Extra acceleration during night (increase rate of product code calls)
  - Reliable timing (stepper motor control, PWM for dimming, …)
    - Replace USB-6251 → cRIO

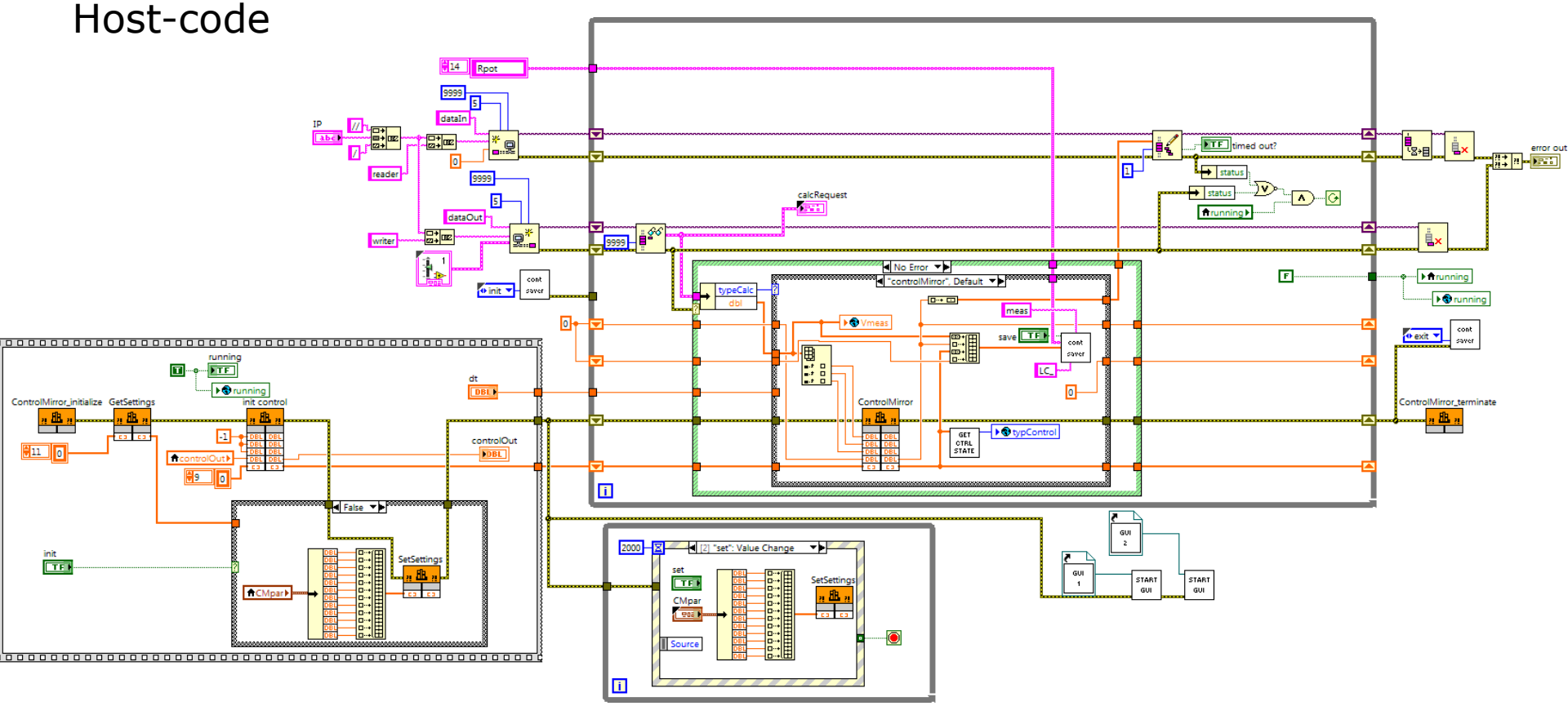
USB-6251 → cRIO implications

- (Old cRIO)
- Different I/O types
- No use of DLL
  - There are alternatives, but no experience (and less integrated building capabilities)
  - Solution
    - cRIO requests calculation to PC
    - PC does the calculation, using DLL as before
    - PC sends result to cRIO

      → not only for control-code, but also for required "sun position"
    - Extra advantage (compared to internal computations)
      - More&easier debugging capabilities

Demo code

Host-code



Demo setup

Summary

- Choosing the best tools for the best (sub-)tasks
  - "Best" is combination of
    - Capabilities
    - Availability
    - Experience
- NI-hardware + LabVIEW
  - Useful for the "hardware part"
- Flexible to change from one type of hardware (DAQ, PC-controlled → realtime hardware)
- Some parts difficult to include in cRIO, using PC (breaking part of realtime-capability) – still easy to do

Questions?

- Questions?

Remarks?