* Introduction
  + Overview
    - APP4MC
    - A4MCAR
    - Problem and Solution
  + Motivations:
    - Importance of parallel programming
    - Evolution of algorithms, applications, and application domains.
    - Evolution of new trends in parallel computing: RTOS, distributed and shared memory communication libraries: OpenMP, MPI, thread-level libraries: Java threads, Python threading, POSIX Threads (Pthreads)
    - Evolution of multi-core processors
    - Embedded multi-core design platform in open-source manner is exciting!
  + Goals and Contributions:
    - Achieving a base platform for multi-threaded and multi-process development that is schedulable and traceable
    - Developing scripts / tool support for several platforms including Linux for profiling, tracing and modeling in order to make demonstration with APP4MC easier.
    - Researching to achieve better software utilization through APP4MC in order to get lower energy consumption.
    - Finding out evaluation metrics and methods for parallel software evaluation using real-time and offline (using e.g. TraceCompass) approaches.
    - Evaluation of several software distributions to make experiments regarding parallel software as well as to test and validate the performance of APP4MC.
  + Events and Publications:
    - IDAACS’17 Paper “Constrained mixed-critical parallelization for distributed heterogeneous systems”
    - EclipseCon conferences Europe 2016 IoT Day, and France 2017 Unconference.
    - Eclipse Foundation GmbH Hackathons I and II
    - International Research Conference Dortmund 2017 Demonstration
    - Google Summer of Code 2017 Participation
* Basics
  + Definitions
  + Design Techniques in Parallelization
  + Synchronization
  + Scheduling Analysis
  + Optimization
* APP4MC
  + What is APP4MC?
  + Design Flow: Modeling, Partitioning, (Prepartitioning), Mapping,,, Algorithms used: ESSP, CPP, ILP, GA
* Design and Implementation of A4MCAR
  + Overall Look, Hardware overlook, Safety overlook,
  + What is the purpose? What is there?
  + Basic Infrastructure, XMOS, RPI/Linux architecture
  + Development Tools: xTIMEcomposer, Linux Shell / Emacs, Eclipse (is a possibility), Android Studio
  + Low-level applications
    - Actuation: Acceleration, Steering, Braking
    - Sensing: Sonar sensing
    - Lighting system
    - Bluetooth comm over uart
    - Ethernet comm (as server) with high-level module (client)
    - Core monitoring
  + High-level applications
    - Schedulable and traceable processes and threads
      * How to trace them?
    - Core monitoring
    - Ethernet client app
    - Camera stream
    - Image Processing app
    - Dummy loads: What is the purpose? To stress Raspberry Pi
    - Dummy graph: What is the purpose? To demonstrate partitioning
    - Touchscreen Display app
      * Aprocess class
  + Web Interface
    - Overall insfrastructure
    - Overall design of the web interface
  + Android Application
    - Looks
    - Improvements to the existing app
* Efforts Toward Tracing
  + APP4MC precise-modeling requires information from the created runnables/threads/processes.
  + Getting instruction sizes of runnables/threads/processes
  + Monitoring processes and threads
  + Taking System traces, trace conversion and visualization: perf, lttng, kernelshark, TraceCompass
  + Created scripts
  + How online timing analysis works in A4MCAR ?
* Efforts Toward Software Distribution
  + Low-level software distribution
  + High-level software distribution
    - From text file..
* Evaluation and Results
  + Limitations
  + Created models
  + Partitioning and mapping details: what is the setup? What is used?
  + High-level Distributions
  + High-level Comparison between distributions
  + Low-level Distributions
  + Low-level Comparison between distributions
* Conclusions
* Thanks
* Bibliography