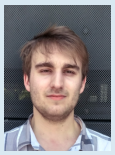


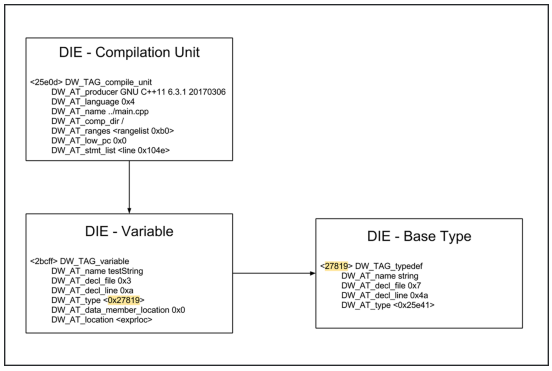
Adrian Fröhlich

Samuel Krieg

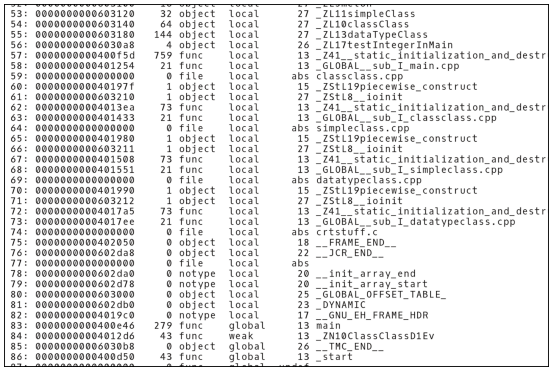
Studenten/-innen	Adrian Fröhlich, Samuel Krieg
Dozenten/-innen	Prof. Stefan Richter
Co-Betreuer/-innen	Christof Gutscher
Themengebiet	Software Engineering - Core Systems
Projektpartner	ABB Schweiz AG , Turgi , AG

Werkzeug für das Debuggen eingebetteter Systeme

Graphische Ausgabe von Variablen anhand von DWARF-Informationen



Graphische Darstellung des Zusammenhangs von DWARF-Daten



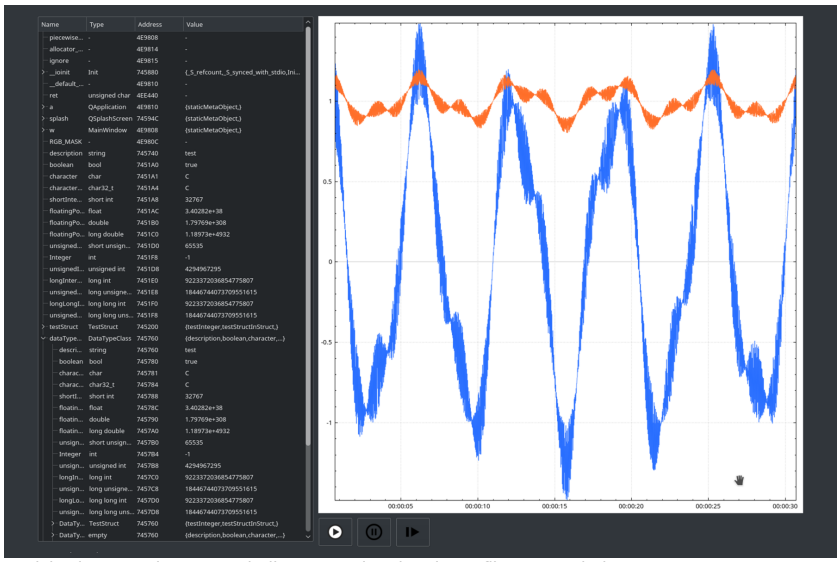
```
53: 0000000000603120    32 object  local     27 _ZL11simpleClass
54: 0000000000603140    64 object  local     27 _ZL10classClass
55: 0000000000603180   144 object  local     27 _ZL13dataTypeClass
56: 00000000006030a8     4 object  local     26 _ZL17testIntegerInMain
57: 0000000000400f5d   759 func    local     13 _Z41__static_initialization_and_destr
58: 0000000000401254    21 func    local     13 _GLOBAL__sub_I_main.cpp
59: 0000000000000000     0 file    local    abs classclass.cpp
60: 000000000040197f     1 object  local     15 _ZStL19piecewise_construct
61: 0000000000603210     1 object  local     27 _ZStL8__ioinit
62: 00000000004013ea    73 func    local     13 _Z41__static_initialization_and_destr
63: 0000000000401433    21 func    local     13 _GLOBAL__sub_I_classclass.cpp
64: 0000000000000000     0 file    local    abs simpleclass.cpp
65: 0000000000401980     1 object  local     15 _ZStL19piecewise_construct
66: 0000000000603211     1 object  local     27 _ZStL8__ioinit
67: 0000000000401508    73 func    local     13 _Z41__static_initialization_and_destr
68: 0000000000401551    21 func    local     13 _GLOBAL__sub_I_simpleclass.cpp
69: 0000000000000000     0 file    local    abs datatypeclass.cpp
70: 0000000000401990     1 object  local     15 _ZStL19piecewise_construct
71: 0000000000603212     1 object  local     27 _ZStL8__ioinit
72: 00000000004017a5    73 func    local     13 _Z41__static_initialization_and_destr
73: 00000000004017ee    21 func    local     13 _GLOBAL__sub_I_datatypeclass.cpp
74: 0000000000000000     0 file    local    abs crtstuff.c
75: 0000000000402050     0 object  local     18 __FRAME_END__
76: 0000000000602da8     0 object  local     22 __JCR_END__
77: 0000000000000000     0 file    local    abs
78: 0000000000602da0     0 notype  local     20 __init_array_end
79: 0000000000602d78     0 notype  local     20 __init_array_start
80: 0000000000603000     0 object  local     25 _GLOBAL_OFFSET_TABLE_
81: 0000000000602db0     0 object  local     23 _DYNAMIC
82: 000000000040119c0    0 notype  local     17 __GNU_EH_FRAME_HDR
83: 0000000000400e46   279 func    global    13 main
84: 00000000004012d6    43 func    weak      13 _ZN10ClassClassD1Ev
85: 00000000006030b8     0 object  global    26 __TMC_END__
86: 0000000000400d50    43 func    global    13 _start
```

Auszug aus der «Symbol Table» des ELF-Files

Ausgangslage: Auch in der Entwicklung von eingebetteten Systemen finden die in der Softwareentwicklung gängigen Testpraktiken Verwendung. Um die Testumgebung von Mittelspannungs-Frequenzumrichtern der Firma ABB zu vervollständigen, soll ein Werkzeug geschaffen werden, welches das manuelle Testen des eingebetteten Systems durch Debugging von Symbolinformationen in einem Emulator unterstützt. Ziel der Arbeit war es einen Demonstrator zu erstellen, der die Machbarkeit des Ansatzes aufzeigt. Der Demonstrator soll in der Lage sein seine eigenen Symbolinformationen auszulesen. Um die Integration in die Testumgebung einfach zu gestalten, sollte es möglich sein, durch geringe Anpassungen am Demonstrator ebenfalls externe Quellen zu analysieren.

Vorgehen/Technologien: In einem ersten Schritt wurde das Verständnis für die Dateiformate ELF und DWARF erlangt. Mit dem gewonnen Wissen wurde die Implementierung dieser Formate in den Compilern analysiert. Dies ermöglichte eine geeignete Bibliothek zu finden, welche beim Auslesen der Dateiformate hilft. Anschliessend wurde die Applikation zum Auslesen von Symbolinformationen mit einer grafischen Benutzerschnittstelle realisiert.

Ergebnis: Es wurde eine Applikation erstellt welche das Auslesen und Darstellen ihrer eigenen Symbolinformationen durchführt. Die Symbolinformationen werden eingelesen, verarbeitet und in einer Baumstruktur hierarchisch geordnet mit Namen und Wert ausgegeben. In einem Graphen kann der zeitliche Verlauf der Symbolwerte verfolgt werden. Da die Anzahl an Debugging Information Entries bei grösseren Projekten sehr hoch ist, gelang es trotz Optimierung nicht, die erwünschte Verarbeitungszeit zu erreichen. Es wurde sichergestellt, dass durch geringe Anpassungen die Applikation ebenfalls Symbolinformationen von externen Quellen verarbeiten kann.



Ansicht des generierten Symbolbaumes mit Echtzeit-Grafik von Symbolwerten